Relationship between Astronaut Head Motion and Space Motion Sickness on Spacelabs 1 and D1.

by

Ilya Shubentsov

Submitted to the Department of

Aeronautics and Astronautics

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science in Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

August 22, 1989

©Ilya Shubentsov 89

The author hereby grants to M.I.T. permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author _____

Department of Aeronautical and Astronautical Engineering

August 22, 1989

Certified by _____

Dr. C.M. Oman, Senior Research Engineer

Thesis Supervisor

Accepted by _____

Professor Harold Y. Wachman, Department of Aeronautics and Astronautics

Chairman, Department Graduate Committee

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEP 29 1989

LIBRARIES

Aero

WITHDRAWN
M.I.T.
LIBRARIES

Abstract

Relationship between Astronaut Head Motion
and Space Motion Sickness on Spacelabs 1 and D1.
by
Ilya Shubentsov
Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the Requirements
for the Degree of Master of Science in Aeronautics and Astronautics
August 22, 1989

This thesis analyses the correlation between astronaut head motion and Space Motion Sickness. It builds on the previous work of R.K. McCoy, who analyzed head motion on the NASA Space Shuttle SL1 mission flown in November-December 1983. On this mission head movements and SMS symptoms and signs were monitored in 2 astronauts. Crew members wore an accelerometer package on the back of their heads. Three axes of angular and linear head accelerations, each sampled at 100Hz, were recorded on Stanford Research Institute eight channel digital Cassette Data Tape Recorders (CDTR). Postflight, this data was read into a VAX computer using a playback unit. It contained a very large amount of playback errors – 10% average, peak as high as 100%. On Spacelab D1 mission (November 1985) similar data was taken from two more astronauts.

Some of McCoy's methods and programs are used for the final analysis. However some changes are made. Among them is a program to detect and classify CDTR errors and, where possible, recover the data. The majority of the unmarked errors were detected and, therefore, the data accuracy was greatly improved. Both Spacelab 1 and D1 data was preprocessed by the error correction program and then analysed.

Of the four subjects, the two who experienced sustained prolonged discomfort show statistically signifficant decrease in activity early in a mission, as expected. The subject experiencing very mild symptoms during the first day only, did not show the decrease in head activity early in the mission, also as expected. The subject who experienced discomfort only during two vomiting episodes did not show a decrease in head activity early in the mission.

The level of the head activity after SMS symptoms subside differed between subjects, presumably due to intrinsic differences in personal style of head movement. The rank order of these activity levels correlated exactly with the rank order of intensity of SMS symptoms experienced earlier in the mission.

Thesis Supervisor: C.M.Oman
Title: Senior Research Engineer

The author gratefully acknowledges the advice and assistance of Dr. C.M. Oman who guided this research and offered help when needed.

The author expresses sincere appreciation to Dr. A. Natapoff for his help with the statistical questions.

The author thanks the four astronauts, who made the data for this research possible; Fred Amlee, Don Harris, and others at NASA/JSC who made this data available; Bill Mayer in the Center for Space Research for his help with equipment questions.

Thanks to Nick Groleau for proofreading the draft and making suggestions.

Big thanks to Kris Mahabir who helped with this thesis, and stoped me from going crasy.

The author thanks all the people at MVL who helped with this thesis or simply made life more fun.

Finally, the biggest thanks go to my parents, who made this researcher possible.

This research was funded by NASA contracts NAS9 15343 and by NAS9 17371.

Contents

Chapter 1

Introduction

Space Shuttle Spacelab mission D1 was flown in November 1985. On that mission as part of the MIT–Canadian Vestibular Experiments[7,8], the relationship between astronaut head movement and space motion sickness (SMS) was studied. Head accelerations, symptoms and signs of SMS were recorded as they occured in two subjects. This thesis presents the results of the analysis of that data. In addition, similar data from the Spacelab 1 mission flown two years earlier and first analysed by R.K. McCoy[1] in his 1985 S.M. thesis, was stripped of errors and analyzed together with D1 data.

1.1 Background

Space Motion Sickness affects the performance, safety and comfort of the crew members. Of primary concern are the first several (2–3) days in space when symptoms are most acute and sensory-motor adaptation to zero-g environment typically takes place. Therefore short duration missions of a week or so are the most affected. However, in extreme cases, symptoms can continue longer, and at least in one case have lasted for 2 weeks[4]. Based on anecdotal reports and evidence from Spacelab1[8] the primary stimulus for SMS appears to be head motion. Visual as well as tactile and other stimuli

also play a role in SMS.

1.1.1 Symptoms of SMS

The symptoms of the SMS are similar to those of motion sickness on earth, e.g. sea sickness, air sickness, etc.:

Pathognomic:	Vomiting
	Retching
	Nausea
	Epigastric discomfort
Cardinal: (almost always seen)	
	Pallor
	Cold sweating
Associated reactions: (frequently seen)	
	Increased salivation and swallowing
	Headache
	Loss of appetite
	Feeling of warmth
	Belching
	Flatulence
	Apathy
	Drowsiness

For a more detailed description of SMS, including etiology, see [2,5].

1.1.2 Incidence of SMS on Previous Flights

SMS has been reported by astronauts almost since the begining of the manned flight. According to [2] and [5] approximately 50% of the crewman of the larger spacecraft on American and Soviet missions experienced SMS symptoms during the first several days.

Spacecraft	Total number of persons flown	Number sick
Vostok	6	1
Mercury	6	0
Voshod	5	3
Gemini	16	0
Soyuz and Soyuz/Salut 1-5	45	11
Apollo	33	12
Apollo/Soyuz Test Project(ASTP)	5	0
Skylab	9	5
Soyuz 25-40/Salyut 6	27	12
Space Shuttle(STS 1 - 25)	85	57

Some difficulty exists in documenting the SMS, however. Originally, when the size of the spacecraft was too small to allow unrestricted motion, there were no reports of SMS. Later, although the spacecraft size increased, documenting the SMS was still complicated because astronauts were reluctant to report it, and because standard reporting methods were not used.

1.1.3 Effects on Performance and Safety

SMS does degrade average efficiency of the crew. On Skylab, the additional time required to complete tasks due to SMS and unfamiliarity with zero-g was estimated at 25%. Crew members appear to respond well to short periods of emergencies or high workload. There are also some cases when safety is an issue, e.g.: extra vehicular activity, in particular vomiting while wearing a space suit – there is a danger the suit life support system will shut down. In particular, in the present suit design both primary and secondary oxygen supply systems may shut down.

1.2 Previous Experiments on Space Shuttle Spacelab 1 Mission

On the SL1 mission (November 28 - December 8, 1983) head accelerations were recorded over a significant portion of the mission by two Payload Specialist subjects. This data was taken using head mounted accelerometers and recorded on a belt mounted digital Cassette Data Tape Recorder(CDTR). The accelerometers, together with the CDTR, were referred to as an Accelerometer Recording Unit (ARU). The subjects also recorded their symptoms of SMS as they occured, using a pocket voice recorder. After the mission a total of 21 CDTR tapes were played back through the NASA LSLE CDTR playback unit and analyzed on a VAX computer at NASA JSC by R.K. McCoy[1]. Unfortunately the record and playback process introduced errors into the CDTR data, most of which were flagged by the playback unit. The word error rate of the data turned out to be about 10% on average. This was more than an order of magnitude higher than the originally anticipated rate of 1 erroneous word in 1500. As a result, kinematic techniques could not be used for the analysis of accelerometer data as originally planned. In his thesis, R.K. McCoy computed acceleration magnitude histograms over successive fifteen minute intervals. He defined the standard deviation of the histograms as activity indices for each axis – linear, angular, and also defined a combined index. Cross plots of discomfort index vs. activity indices were constructed. The following sections review the design of the flight experiments and methods used to collect the data, the characteristics of the ARU and CDTR playback unit, and a summary of R.K. McCoy's methods and results. Readers needing additional details on the SL1 experiment are referred to McCoy's thesis[1].

1.2.1 SL1 Experimental Methods

Pre-flight Training

Four payload crewmember subjects (alphabetically coded A through D) were extensively trained prior to the mission [2]. Classroom training consisted of 50 hours of lectures on vestibular physiology, spatial orientation and motion sickness[3]. They were also trained in recognition of SMS symptoms while wearing left/right vision-reversing goggles, and in using the ARU. Additional training took place during the general motion sickness susceptibility tests.

In-flight Experiments

Two of the four crew members wore the ARU extensively throughout the mission. They also recorded their SMS symptoms on pocket voice recorders. The crew members were instructed to record the symptoms at regular intervals, and whenever they occured or changed significantly. Although the importance of detailed reporting was emphasized, often subjects could only provide a short report consisting of a single number. This was a numerical magnitude estimate of the intensity of the overall discomfort, using the method developed by Bock and Oman [9]. Instructions to the subjects were "Pick a sensation magnitude of overall discomfort in the middle of the 'moderate' range, halfway to vomiting. Call this standard '10'. Estimate the magnitude of overall subjective discomfort with respect to it. If no sensation, say 'absent'. If just noticable, say 'threshold'." Because the overall discomfort reports were made relatively frequently, they formed the basis for correlation with the head movement data in McCoy's thesis and in this thesis.

When time permitted, subjects made longer and more detailed reports using a check list shown below:

TIME (dd/hr/min)
OVERALL DISCOMFORT (Ratio)
EPIGASTRIC AWARENESS (location)
OR DISCOMFORT (location, Slight/Moderate/Intense)
NAUSEA (Ratio)
HEAD MOVEMENTS (which axes?)
COLD SWEATING (location, S/M/I)
SUBJECTIVE WARMTH (S/M/I)
SALIVATION
DRY LIPS
RESPIRATION
HEADACHE (location, S/M/I)
DROWSINESS
YAWNING
BELCHING
FLATULENCE
DIZZINESS/DISORIENTATION (describe)
APATHY
CONCENTRATION (impaired)
APPETITE
SENSITIVITY (to sensory stimuli)
PALLOR (S/M/I or Yes/No)
FLUSHING
FLUID SHIFT–FACE (eyes, jowls, veins, stuffiness, fulliness)
VOMITING, RETCHING (time, duration, relief after?)
DRUGS (which, when, side effects, effectiveness?)

Experimental Equipment - ARU

Astronauts' head movements were recorded with an occipitally mounted package of linear and angular accelerometers, connected to a belt mounted power pack and the CDTR. Subminiature piezoresistive linear accelerometers were used for the X, Y, and Z axes (axes are shown in Figure 1.1, taken from [1]). They were chosen partially for their small size, weight, and power consumption[1]. Their resonant frequency (between 500 and 1000Hz) was much higher than the expected head motion range or the cutoff frequency of the antialising filter in the CDTR. These accelerometers had a maximum

[1]Kulite Accelerometers model GY-125-20. 0.13x0.15x0.30 in and .25 gram

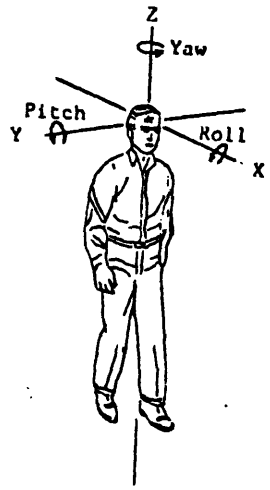

Figure 1.1: Accelerometer axis directions on Spacelabs 1 and D1

transverse sensitivity of 3%. Their sensitivity was chosen as 0.125 V/G.

Closed loop torque balancing servo accelerometers were used to record the angular accelerations [2]. Their sensitivity was 60 Rad/sec^2/V and cross axis sensitivity 2%. These accelerometers also have a very low temperature sensitivity – 0.002%/C° thermal coefficient at 0C°, and the operating range of -40C° to 95C°. On Spacelab 1 the accelerometer package was held at the back of the subjects head by an adjustable cloth head band.

A special purpose Cassette Data Tape Recorder (CDTR) was developed for NASA by Stanford Research Institute (SRI), and used to record the analog acceleration data on digital tapes. The CDTR was worn on a waist belt together with a lithium-bromide power module, which could supply power to the ARU for at least 60 hours. The CDTR recorded up to eight channels of data – 6 digital and 2 analog. Each channel was sampled at 100Hz, 10 bit data resolution (over a ±2.5V range) and an 11th even parity bit was added to the digital channels. Up to eight hours can be recorded on

[2]Schaevitz Accelerometers model ASM-300. 1 in^3 volume, 2oz.

a special 1/8 inch recording tape cassette. Self clocking Miller encoding (now known as Modified Frequency Modulation – MFM) was used to record the digital data. The CDTR also contained an asynchronous digital clock module which was used to time stamp the data every minute (or every 6000 samples). Time was recorded modulo 12 hours, and converted to Mission Elapsed Time (MET) during analysis based on notes of tape start and stop times made by the crewmembers, and the known offset between the CDTR clock and MET.

On SL-1 only the digital CDTR channels were used:

Channel	Contents
1	X acceleration
2	Y acceleration
3	Z acceleration
4	Analog 1 – not used
5	Roll acceleration
6	Analog 2 – not used
7	Pitch acceleration
8	Yaw acceleration

Data Processing Equipment - CDTR Playback Unit

The data cassettes were played back after the mission through the SRI playback unit interfaced to a VAX 11/780 computer with a 10 bit direct memory access parallel interface (DR11W). The software used to transfer the data to a RP06 disk was the program PB.EXE[6]. Below is the format used to store data on the VAX:

Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word Format -	T_2	T_1	T_0	R	E	P	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Word Definitions:	D_9-D_0	10 Bit Cassette Data Word D_9–MSB D_0–LSB														
	P	Recovered parity bit for cassette data word. Even parity used.														
	E	Data error bit. No error–0, Data word error–1														
	R	R wave detected, used only with ECG data channel, not used on these missions.														
	T_2,T_1,T_0	Track number – 3 bits. Specify 1 of 8 tracks 001 = Track 1 010 = Track 2 011 = Track 3 100 = Track 4 101 = Track 5 110 = Track 6 111 = Track 7 000 = Track 8														

Files were labeled S1-S32 corresponding to the last 2 digits of the tapes serial number provided by the manufacturer.

The CDTR playback unit had some error detection capability – it would set the data error bit if it was unable to properly decode the Miller encoded signal or if it detected an even parity error. Often the difficulty occurred because the playback unit was unable to reconstruct the CDTR Miller clock properly. Other sources of error included tape skew, bounce and wrinkles in the tape medium. R.K. McCoy relied on the accuracy of the error detection scheme to exclude erroneous data from analysis. He also excluded three data values immediately prior to an error, since SRI suggested there was a significant chance that they would also be in error.

1.2.2 R.K. McCoy Data Analysis on SL1 Mission

Error Analysis

When the SL1 mission data was played back and then analyzed for errors, McCoy found that the error rate was much higher than specified by SRI. The average word error rate was about 10%. Some of the errors were "soft" while others were believed to be "hard" errors. Soft errors may be eliminated by careful adjustment of the playback

unit. Hard errors represent the recording errors in the CDTR and cannot be recovered during playback.

Data Analysis

Due to the high error rate, a statistical method was used to analyse the data. Acceleration magnitude histograms for 15 minute intervals were computed. The 15 minute length of this interval was chosen because it provided a sufficient amount of data without too much detail and variability, and was consistent with the 10 to 45 minute typical time course of the development and remission of motion sickness on Earth. It also helped to eliminate accelerometer drift effects from the data analysis. Accelerometer bias changed very little over 15 minutes, but may change substantially over 8 hours.

The standard deviations of the acceleration histograms were computed and referred to as the "Activity Indices" for each of six axes. Units were G's for the linear accelerations and Rad/sec^2 for the angular accelerations. A "Linear Index" and an "Angular Index" were also computed by adding together the three separate linear and angular indices, respectively. Finally, an ad-hoc "Total Activity Index" was also computed by adding the "Linear Index" and the "Angular Index" using a weighting factor. This factor was obtained by dividing the mean of the Linear Index by the mean of the Angular Index for all of the data analyzed from the mission, and was then used to scale the Linear Index. This was done to approximately balance the contribution of the angular and linear motions in the Total Activity Index.

1.2.3 Results of McCoy's Statistical Analysis

It was found that the subjects' activities increased significantly after MET day 3. Overall the data was consistent with the hypothesis that subjects reduced their head

accelerations when sick. Below is the table summarizing the means of activity indices, by subject, for days 0–3 and 4–6.

Subject	MET days 0-3		MET days 4-6	
	Linear index	Angular index	Linear index	Angular index
B	1.06G	9.12Rad/sec^2	1.86G	12.38Rad/sec^2
C	3.38G	8.19Rad/sec^2	5.07G	9.92Rad/sec^2

Using a t-test these means were found by McCoy to be statistically different to at least .005 significance level. Also, both subjects reported less motion sickness after MET day 3.

Also an inverse relationship between the activity indices and the discomfort ratio was found for subject B (there was not enough data for subject C to make any firm conclusions).

1.3 Experiments on Space Shuttle Spacelab mission D-1

The Spacelab 1 vestibular experiments were flown again in November 1985 on the first German Spacelab, D1. This was a seven day mission on the Space Shuttle Challenger.

1.3.1 Pre-flight Training

The subjects were five payload crew members, alphabetically designated F through J. All five subjects received extensive training on recognition and reporting the symptoms of SMS. Training included classroom instructions, provocative stimulation (with left-right vision reversing goggles), Coriolis head movements, and KC-135 parabolic zero-g flights.

1.3.2 In-flight ARU Experiments

Two male crew members, designated I and J, wore the ARU during the mission. They did not use anti-SMS drugs during the mission. In contrast to Spacelab 1 where both ARU wearing subjects were frankly sick, on this mission only subject I experienced severe symptoms. Reporting instructions were essentially unchanged from SL1 (see above).

However, on the D1 mission both subjects were also asked to wear a foam neck collar for several hours. The reason for wearing the neck collar was to see if it would significantly reduce the subjects head motion. Head motion, in particularly rotational motion, was hypothesized to be the most provocative SMS stimulus. If astronaut head motion could be reduced by wearing the neck collar to at least the same level exhibited when they are sick, then perhaps the incidence of the SMS could also be reduced. To see what reduction of motion could be achieved, each subject was asked to wear the neck collar late in the mission for three hours when their ability to move was not otherwise limited by symptoms.

1.3.3 ARU Wearing Coverage

The ARU wearing time coverage was not as extensive on this mission as it was on Spacelab 1. However, reasonably detailed acceleration and discomfort data during the first two days was obtained. Sufficient data was collected in the later stages of the mission for determination of the motion indices. Unfortunately the discomfort data did not substantially overlap the ARU data for subject L during the first few days of the mission. Also, the discomfort index was infrequently reported by both subjects after the first few days of the mission[3].

[3]This is partially explained by the fact that both subjects did not experience SMS after the first 2-3 days in space.

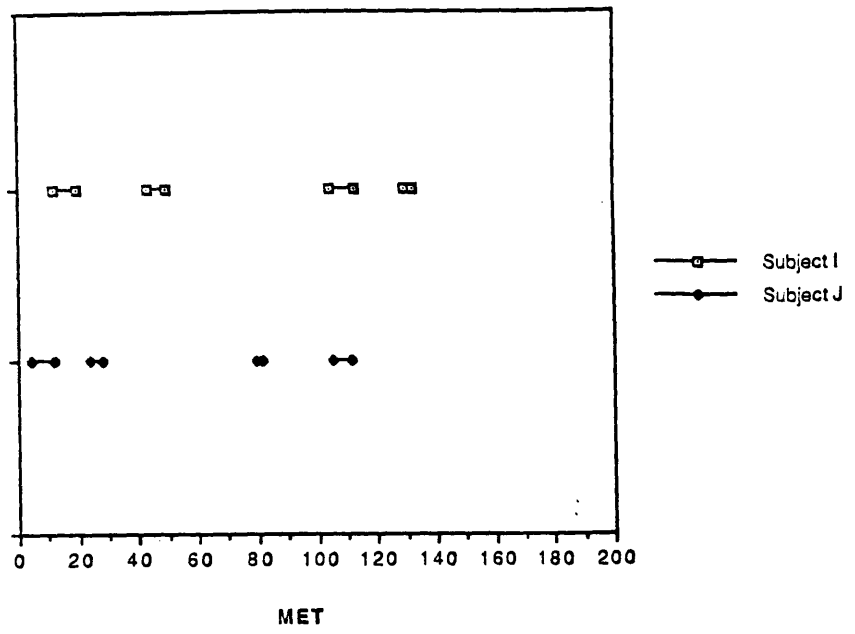

Figure 1.2: ARU wearing coverage on Spacelab D1 mission

The total amount of ARU data for subject K was 26 hours, and for subject L was 12 hours, see Figure 1.2 below. In contrast, on Spacelab 1 mission subject B wore the ARU for about 88 hours, and subject C for about 49 hours.

1.3.4 D1 Experimental Equipment

The experimental equipment used was similar to that used on the SL1 mission (described earlier). The changes were:

- The ARU accelerometers were mounted on a "snoopy"-style pilots hat, which were more comfortable to wear than the SL1 head band.
- Different linear accelerometers were used[4] – the gain was 2G/V, with -2.5V to +2.5V range. The accelerometers were physically larger models, so the head

[4]Setra Systems model 141a

package was slightly larger in size.

- All 8 CDTR channels were digital.
- Two CDTR digital channels were added and used to record data from skin temperature and pallor probes. Below is the ARU channel assignment on the D1 mission[5].

Channel	Contents
1	X acceleration
2	Y acceleration
3	Z acceleration
4	Skin temperature
5	Roll acceleration
6	Skin pallor
7	Pitch acceleration
8	Yaw acceleration

- Orthopedic style foam cervical collars were provided

Data Playback Unit

As on SL1, the data was played back on a SRI playback unit interfaced to a Vax computer. The same physical data playback unit was used as for the SL-1 data, with somewhat modified software. Tapes were inspected for quality prior to flight. More attention was paid to adjusting the playback recorder heads to reduce the error rate. Files were recorded on RA60 disk packs and Exabyte 8mm tapes and transferred to a MicroVaxII at MIT for further analysis. Files were labled D1SN1091.DAT, D1SN1093.DAT, D1SN1094.DAT, D1SN1109.DAT, D1SN1110.DAT, D1SN1114.DAT, and D1SN1125.DAT based on the cassette serial number.

[5]Some engineering documents show channels 4 and 6 switched. Assignments shown are believed correct based on inspection of the actual pallor and temperature data

1.4 Hypothesis/Objective

The objective of this thesis was to more effectively eliminate errors from CDTR data and to analyse the data from both SL1 and D1 missions. The main hypothesis of this thesis concerns the dependence of the discomfort index and the head motions during the early and late stages of a mission. Specifically, it was hypothesized that the the subjects experiencing SMS symptoms voluntarily reduce their head motion during the adaptation period in the first several days of the mission (until the SMS symptoms subside). Conversely, the subjects who are not experiencing SMS should show no difference in activity levels. It was also hypothesized that susceptibility to SMS depends on the astronaut's style of head movement as reflected in the average level of angular head acceleration when asymptomatic. Specifically, the higher that level is the more likely an astronaut is to experience SMS.

Chapter 2

Experimental Methods

The head acceleration histograms were computed using R.K. McCoy's programs [1]. The data analysis path is shown schematically in Figure 2.1.

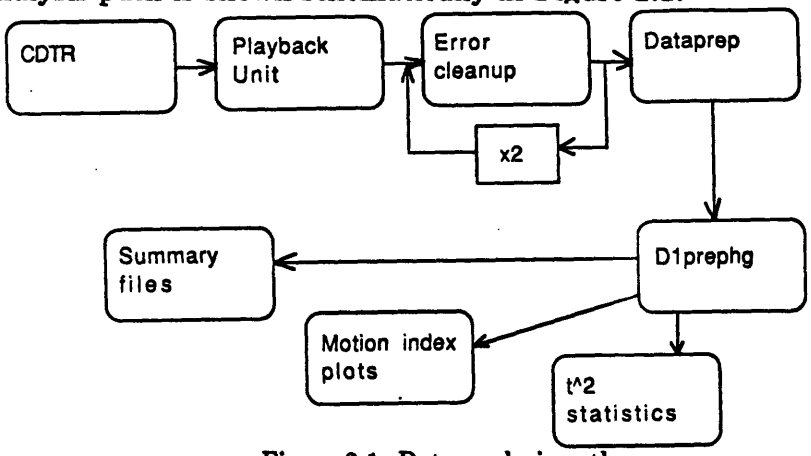

Figure 2.1: Data analysis path

For the D1 analysis, McCoy's programs were modified to account for different settings of linear accelerometer gain. Additional modifications were made and utility programs were written for formating and exporting the data for graphing. The main differences between McCoy's analysis of Spacelab 1 data and the results in this thesis are:

- The data was preprocessed by ERROR_CLEANUP, an error detection and classification program, which was developed for, extensively tested, and used in this

thesis.

- A somewhat different motion index was defined and used to measure subjects activity levels.
- Variability of the data was computed, and Chebyshev's inequality[10] was used to establish the statistical significance of changes in the motion index
- Controlled motion activities were identified and separated for purposes of analysis.
- McCoy excluded files with high error rate from his analysis. In this thesis all D1 files were analysed. Some Spacelab 1 data was excluded, but only those parts of the files that contained very high error rates (see File Index appendix for details).

The use of ERROR_CLEANUP was needed because the data was found to contain unmarked errors. Normally, if a data value is incorrect the CDTR error flag for that value is set. However, there was some data for which this clearly did not happen. In addition, some seemingly correct data was marked as in error. After examination of large portions of the data manually, the patterns of errors were noticed. Then a VAX program, ERROR_CLEANUP, was written to classify errors and set or clear the error flags, and in a few cases even correct the data. This method was not fool proof, but it appeared to set the error flag on most of invalid data, and clear it on most of the valid data, and was believed unlikely to introduce any systematic bias into the results [1]. The program is written in VAX Fortran version 4, operating system VMS version 4.4. It was used with VMS version 4.5 and 4.6 without any problems. No special software is needed to run this program [2].

[1] See Error Detection, Classification and Recovery chapter for more details

[2] However, enough disk space to hold the results is required – the program stops when it encounters writing errors on output

There were also other differences in analysis of data. On D1 the ARU was not worn as long as on SL1, particularly when the crew was symptomatic. Because of this a cross plot of accelerations vs. discomfort was possible only for small portions of the data. Also, the linear acceleration data, though processed, was not used in the analysis. This was done for three reasons:

- Signal gain was found to be too low to give sufficient resolution to the data – typical head movements produced very small changes in recorded acceleration data. McCoy probably did not note this because SL1 data was corrupted by errors.

- It was felt that much of the linear acceleration data was due to head rotation, not translation. Manual examination of the data showed a very high degree of correlation between high linear accelerations and high angular accelerations. Although the linear accelerometers were placed in the best possible location, they were still several inches away from the center of rotaion. Therefore, this resulted in increasing the contribution of angular head motion to the linear acceleration data.

Individual 15 minute activity indices – standard deviations of the acceleration histograms – were used to compute a somewhat different motion index (instead of McCoy's "total activity index"). It was defined as:

$$M = \sqrt{\frac{N_r A_r^2 + N_p A_p^2 + N_y A_y^2}{N_r + N_p + N_y}} \tag{2.1}$$

where M is the motion index, A_r, A_p, A_y are roll, pitch, and yaw activity indices respectively (i.e. histogram standard deviations as used by McCoy), with N_r, N_p, N_y the number of valid data points in each channel. One of the reasons for using a different

index is that it is inappropriate to simply average the standard deviations across time or axes. That is because the average would depend on the grouping in time of the original data and would be different from the standard deviation computed from the entire original data. By squaring and weighting the activity indices, the same motion index is obtained as would be computed directly from the entire set of the original data.

Angular activity indices for all three axes are used to compute the motion index. Thus if the error rate is a lot lower in one of the axes, there will be more measurements for this axis and therefore it will be more heavily weighted in the the total motion index. This was not a problem for D1 data.

The data was grouped naturally by ARU sessions. A session is a time series of acceleration data continuous in time, recorded on a single cassette. However, during analysis these ARU session data may have gaps in the time series either due to high error rates, or because some data was specifically excluded because the subject head movement was constrained.

A t^2 parameter (but not necessarily t^2 distribution) was used to assess the difference between the motion indices for different ARU sessions. The difference was large enough to be statistically significant, independent of the underlying distribution (therefore, it was unnecessary to assume the similarity of variances of different ARU sessions that would rigorously justify the use of the t^2 distribution). X_i $(= M_i^2)$ is the average square deviation of the acceleration from its mean value computed over a particular 15-minute interval. Then the mean for the whole ARU session is given by:

$$\overline{X} = \frac{N_1 X_1 + N_2 X_2 + \ldots + N_n X_n}{N} \tag{2.2}$$

where

$$N = N_1 + N_2 + \ldots + N_n \tag{2.3}$$

The square root of $\overline{X}$ is therefore $\overline{M}$—the motion index for the ARU session.

If A_r, A_p, and A_y are respectively roll, pitch, and yaw linear activity indices for a single 15-minute window, calculated from N_r, N_p, N_y points each, then the individual number $N_i - 1$ (independent points or degrees of freedom) of X_i is taken to be

$$N_i - 1 = \frac{(N_r - 1) + (N_p - 1) + (N_y - 1)}{3} \tag{2.4}$$

This conservative counting is intended to avoid counting these indices for different space axes more than once for each observation time. That is because they are highly correlated, they reflect the same conditions and their associated histograms usually have similar shapes. For convenience these are combined into a single index M and its imputed number of degrees of freedom is counted conservatively.

The variance, S^2, of X_i about $\overline{X}$ is given by:

$$S^2 = \frac{\sum_{i=1}^{n} N_i X_i^2 - N\overline{X}^2}{(N-1)} \tag{2.5}$$

Once N, $\overline{X}$, and S^2 are computed for two ARU sessions (A, B), a tentative t^2 statistic was found from:

$$t^2 = \frac{(\overline{A} - \overline{B})^2}{\left[\frac{(N_A-1)S_A^2+(N_B-1)S_B^2}{(N_A-1)+(N_B-1)}\right]\left[\frac{1}{N_A} + \frac{1}{N_B}\right]} \tag{2.6}$$

where N_A and N_B are the number of 15 minute activity indices used in computing $\overline{A}$ and $\overline{B}$ respectively, taken to be the number of independent measurements of the motion index.

This approach is most appropriate when astronauts are not performing a highly repetitive task. When astronauts perform a single repetitive task that strongly influences their activity indices for more than 15 minutes, a smaller N_x should perhaps be used. It is believed that this situation occured rarely, if at all. The data corresponding to specific activities that physically restricted head motion (i.e. the hop and drop, and dome experiments, see below) were excluded from this analysis.

The t-test assumes that the variances of A and B sessions in the equation for t^2 above are the same. In fact, however, they are apparently different. Therefore the p-value from a t-test (typically $< .0005$ for this data) cannot be used rigorously. However, applying Chebyshev's inequality[10], which does not assume this equality of variances, gives a p-value of no more than $\frac{1}{t^2}$ (typically here $\approx$ .02–.04, which is sufficient to establish a significant difference) no matter what the underlying distribution.

Another change in analysis was the separation of controlled and uncontrolled head motion. Specifically, when computing the motion index, the data corresponding to the hop and drop experiments (see next section) and the dome experiments was excluded. The dome experiment forced the motion index to nearly 0 for its duration, since the astronaut's head was rigidly fixed (through the use of bite board) to the experiment apparatus. Likewise, during the hop and drop experiments the astronauts were forced to increase their motion index.

2.1 Identification of Head Constrained Activities on D1

In addition to the head motion experiments there also were other experiments that constrained astronaut head motion. Among these were hop and drop, the dome, and the neck collar experiments. There were two hop and drop sessions, two neck collar sessions (one for each subject), and 4 dome experiment sessions in which the subjects

participated. This impacted the data in two ways – directly, by constraining the character of the head accelerations, and indirectly if they caused the changes in the discomfort indices.

Hop and Drop Experiments

There were two hop and drop sessions on the D1 mission. The first one occured during the period of adaptation. Subject I recorded severe SMS syptoms, including vomiting, within two hours prior to the time of this session. The second hop and drop session took place well after the adaptation was over. There is very little ARU data for the time around the second hop and drop – only one 15 minute interval per subject (two subjects shared the ARU in time). The symptoms were not recorded during that session, but no severe symptoms were indicated by the subjects.

Subject	MET time			
	Day	Starting time	-	Ending time
F	0	14:10	-	14:30
	6	14:50	-	15:02
I	0	14:32	-	14:47
	6	15:08	-	15:23

The effect of the hop and drop experiments is to dramatically increase the activity indices for the 15 minute intervals. That is because the subjects are instructed to jump up/down while strapped to the deck with elastic bungee cords. The increase is so large that it significantly affects the 8 hour session mean. Therefore, the accelerometer data for the periods corresponding to the hop and drop experiments was excluded from the analysis of the unconstrained head motion.

Dome Experiments

Subject I took part in 4 dome sessions (8 runs). There is no data for subject J, except for one note:

Subject	MET time			
	Day	Starting time	-	Ending time
I	0	17:06	-	17:12
		17:22	-	17:28
	2	14:21	-	14:27
		14:35	-	14:41
	3	14:30	-	14:36
		14:40	-	14:41
	4	12:56	-	13:02
		13:07	-	13:13
J – no experiments?				
J	2	15:26	-	"performing experiment"

The effect of the dome experiments is to lower the motion indices for the 15 minute intervals. That is because the subjects' head is rigidly fixed with respect to the dome using a bite-board. The dome sessions typically were only 6 minutes long and they did not change the 8 hour ARU data much. They should be considered, however, when the variability of the motion index over time is analyzed. This data was also excluded from the "head unconstrained" analysis.

Neck Collar Experiments

Below is the time table of the neck collar experiment.

Subject	MET time			
	Day	Starting time	-	Ending time
I	5	10:00	-	10:34
J	4	06:30	-	09:59

Very little data, much less than expected, was available from this experiment. Subject I wore the neck collar for only half an hour. Subject J did wear the collar for several hours, but unfortunately did not record the head motions with the ARU for most of that time. There is only one hour of ARU data for subject J while wearing the collar.

Chapter 3

Error Detection

3.1 Error Cleanup

All data files were first processed twice with the ERROR_CLEANUP program. Double processing was found to remove errors a little better that a single pass. Extensive spot checking of D1 data was done to ensure that obvious unmarked errors did not remain in the data (time did not permit manual examination of SL1 data). The original data and the cleaned D1 data were compared. Where large differences in error rates and/or data values were found, the data was examined manually.

3.2 Error Classification and Recovery By The ERROR_CLEANUP Program.

The ERROR_CLEANUP program was designed to set the error flag on erroneous data and to clear the flag when it was incorrectly set or when the data values could be corrected by scaling it by a power of 2. This program uses heuristic methods to detect and to classify errors. Different properties of the data are determined and form a basis for the error detection and classification. Criteria based on time histories of data smoothness, running average, and the CDTR error flag status are used to detect an error string and determine its span. Then five parameters were computed for each

string of data that was either suspected of being erroneous, or flagged as such by the playback unit. Error classification was based on these five parameters.

3.2.1 The Types Of Errors

CDTR data was manually examined and "inconsistencies" in the data were classified into four "groups" of errors. Here "error" refers to a string of data that is marked as erroroneous by the playback unit, or is suspected of being an error by the ERROR_CLEANUP program (even if its error flag is cleared). As detailed below, each group in turn contains several "types" of errors. The groups are:

1. Short and possibly "correct" errors. These are the strings of data with the error bit set, but which are suspected of being in fact valid. These are fairly harmless errors, since they are entirely marked by the playback unit. Even if some of them are recovered incorrectly by the program they are not likely to significantly corrupt the data unless present in huge numbers, because they contain reasonable values.

2. Very long error intervals - their error flag is set by the playback unit for typically thousands of values. These errors are also fairly harmless, since they are entirely marked as errors by the playback unit.

3. Short errors that contain obviously invalid data. The error flag on some of the invalid values may not have been set. These are very serious since they may significantly distort the statistics if they are undetected, even if present in small numbers.

4. Error intervals that start with obviously invalid data for which the error flag has not been set. These are also serious since they can distort the statistics and may

be completely undetected by McCoys programs.

5. There is also an "unknown" error group – it contains errors that are not classified as any other group/type. They can be a nuisance since they can take out a very large percentage of data.

This ad hoc classification was chosen because it seemed that most errors were of one of these types, and because each of the groups/types required different handling.This classification is somewhat arbitrary and a different one could have been used. This scheme was chosen after many days of examining the data manually and trying to guess the reasons for errors, knowing some of the failure modes of the playback hardware involved. Also, a fairly large number of error types within groups were used to allow actions to be taken very selectively on different errors.

- The first group contains errors that are not very long, typically 1 to 20 values (= .01 to .2 seconds) in length, and never longer than 200. These error intervals appear to contain correct values that were flagged by the playback unit as incorrect for one or another reason. There are four types of errors in this group. The first one contains slightly varying data values that "behave well" and match the values outside the error interval. "Behave well" means that the data passes a goodness of linear fit test on up to 5 points. (The actual number of points used depends on how many are available. Due to the short length of an error interval, for example, fewer than 5 points may be used. When only two points are available a difference test is used.) The program computes the sum of the squares of the difference between each data point and the linear approximation, then divides this by the degrees of freedom (i.e. the number of points minus one). If the result is less than a smoothness criteria constant, then these points are

considered "smooth". This linear fit test is implied whenever there is a reference in this thesis to smoothness of data.

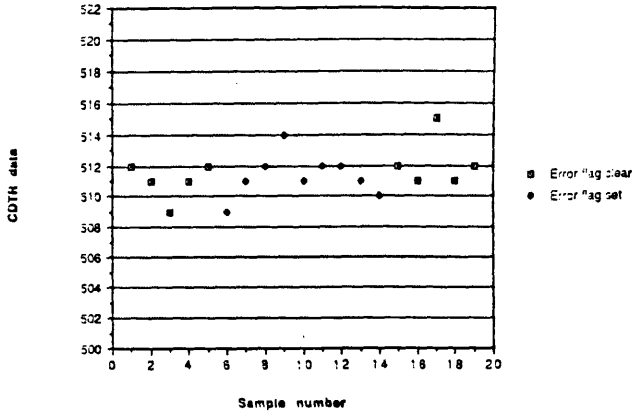

Figure 3.1: Error group 1, type 1

The second type of error contains data values that are constant everywhere but at the very last point—i.e. the error flag is cleared on the next data point.

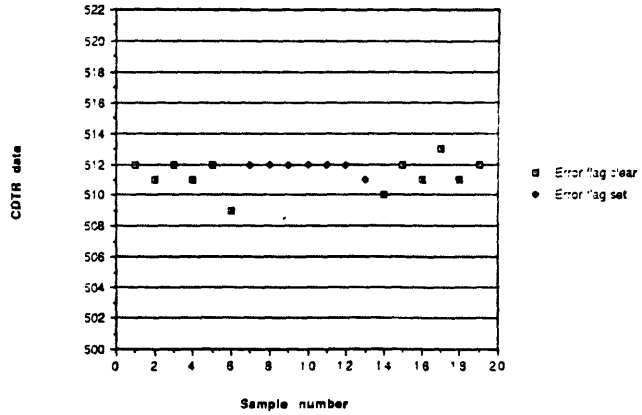

Figure 3.2: Error group 1, type 2

The third type of error in this group is one where the data is entirely constant, but possibly valid. The error flag may be cleared even before the data value changes.

The last error type is similar to the third one, but the data may actually be incorrect. It appears to be abnormally constant.

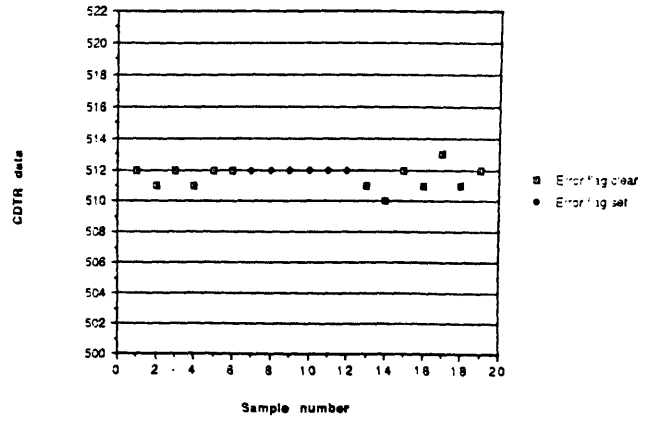

Figure 3.3: Error group 1, type 3

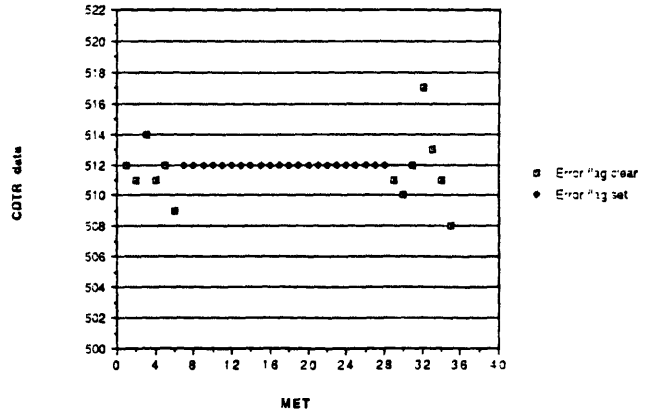

Figure 3.4: Error group 1, type 4

- The second group contains very long error intervals, tens or maybe even hundreds of thousands of points long. Generally, any error longer than 200 is classified as UNKNOWN_ERROR group, unless it happens to be very specific. There are three such particular types. The first one is a very long error with values that are entirely constant. The second is one in which the data is constant until the last value, where its value becomes 1023 (highest possible value on the CDTR). The error flag is then cleared on the next value. The values following the error are all at 1023. The third type of error in this group is a constant error value of 1023. That happens when the error bit changes to "set" on a very long run of 1023.

The third and fourth groups of errors are potentially serious since they, if undetected, will affect the statistics.

- In the first two groups, all the data values are entirely marked as errors by the CDTR playback unit. Some of the errors in the third group are not. The first type of error interval in this group contains erroneous values that are constant for many points, and are sandwiched between apparently correct values. Most of the erroneous data is equal to one of only few, but widely different values. Some of the apparently correct values just before and just after the clearly erroneous data are marked as error. However, some of the erroneous data is not flagged.

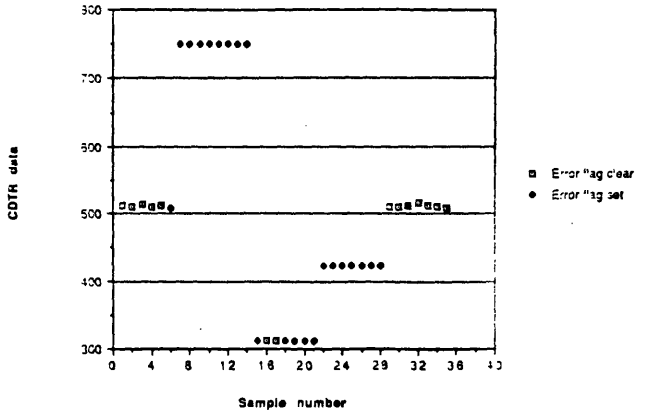

Figure 3.5: Error group 3, type 1

The second type of error in this group contains a few erroneous values between apparently valid data. However, the entire error interval is flagged.

The third type of error is just like the first, but here all of erroneous data has the same value.

The fourth type is just like the third, but it is entirely marked.

The fifth type of error contains runs (both marked and unmarked) of what looks like correct data multiplied or divided by a power of 2 and sandwiched between

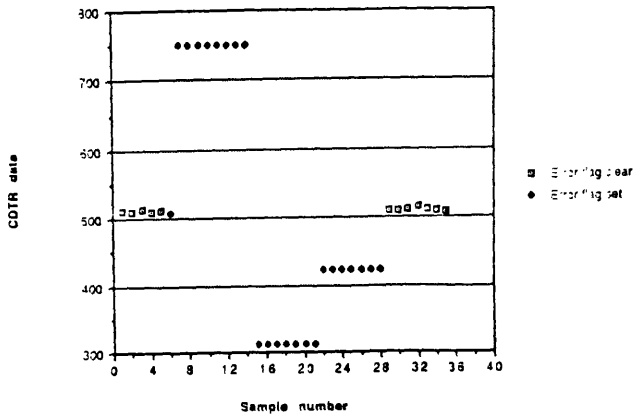

Figure 3.6: Error group 3, type 2

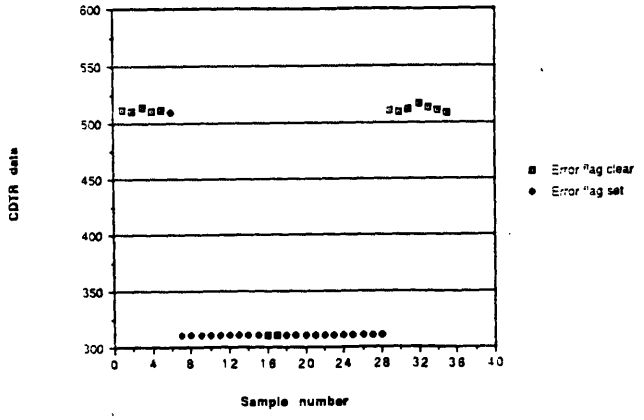

Figure 3.7: Error group 3, type 3

valid data. Such errors probably originate in the shift register of the CDTR playback unit.

- Errors in the fourth group, unlike all others, start unmarked. Normally processing of an error starts when an error flag is encountered. The overwhelming majority of all errors are marked at the start. However, some errors are not. They have to be detected by a smoothness test. There are four types of errors in this group. The first one is an erroneous constant error interval. These error intervals are very short, typically just one point.

  The next type is a data error that is apparently partially shifted by a power of two. This type is correctable. These errors usually occur just before a value with

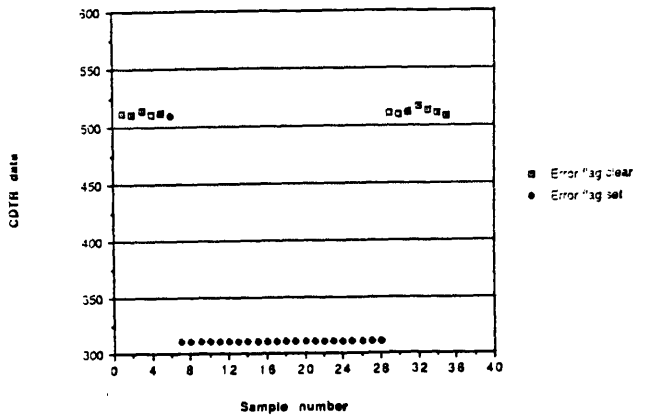

Figure 3.8: Error group 3, type 4

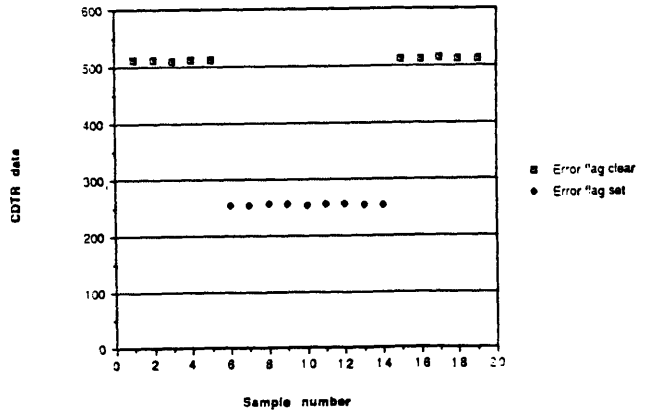

Figure 3.9: Error group 3, type 5

error flag set and are apparently shifted by a power of two.

The third type of error is relatively short—less than 200 points—and not correctable. It includes all short errors in this group with the exception of the first two types. The only reason constant errors were not included in this type was because they are relatively "harmless" because of their length. (This does not mean that they can not distort the data for the whole ARU session – on the contrary, they can if they go undetected. However, the values sufficiently different to distort the statistics are easily detected by a smoothness test.) This way the program keeps track of other non correctable errors in this group separately. The fourth type of error in this group includes all long errors, i.e. longer than 200.

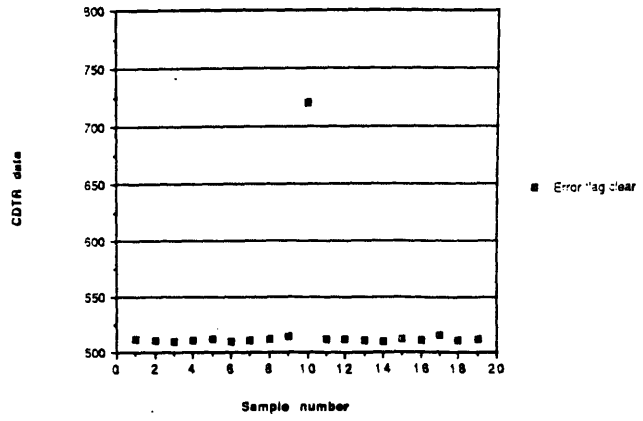

Figure 3.10: Error group 4, type 1

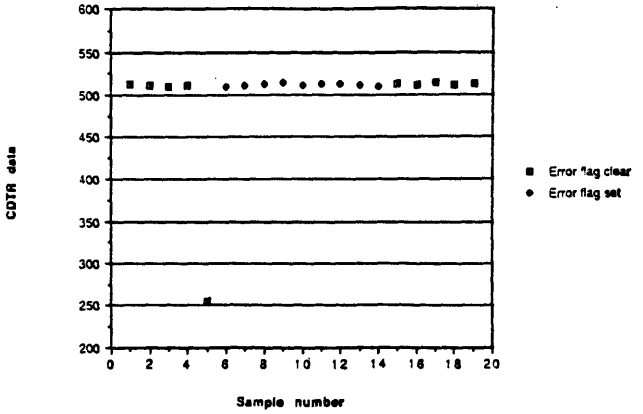

Figure 3.11: Error group 4, type 2

- There is also an "unknown error group". All errors not classified in any other group are labeled as unknown. Notice that the fourth type of errors in the fourth group (the last one described above) would be labeled as "unknown" most of the time by the ERROR_CLEANUP program.

This description of errors is rather ideal in the sense that the program does not always identify the errors the same way a human operator would. This is not really a problem, since the program is conservative—it is designed to take a safer action in dubious cases. When looking at the output of the program, sometimes it will seem like the program is misbehaving, but on closer examination everything is O.K. [1] A human

[1] Do not assume that you found a bug if you see unexpected results.

observer may misinterpret the error type, unless the data is closely scrutinized.

There is a small chance that the program will not set the error flag on erroneous data, or even clear the flag that has already been set on truly erroneous data. There are two things that the user can do to increase the conservatism of the program further (See appendix E for program listings). First, the parameters in the `error_type` subroutine can be set not to clear any error flags. This way no erroneous data will be cleared. The disadvantage of this is that more data will be lost. The second approach is to adjust several parameters for stricter error filtering: `maximum_smooth_jump` and `smoothness_criteria` in the `ERROR_CLEANUP` program, and `light_limit` and `tight_limit` in the `ERROR_TYPE` subroutine. (See description of these in the Appendix A.) The disadvantage of doing this is that some of the perfectly valid data may be declared as erroneous and the error flag will be set. The problem is that only the more variable data will be thrown out and this may change the statistics of the overall data. Hence these parameters should be chosen very carefully. This is a very important point. Many different values of the parameters were tried until the ones that seemed to get rid of real errors, cleared valid data and did a good job in marginal cases were found. They should not be changed unless the gains of the data channels are different from those used on D1 mission. Also, two passes of the Error_cleanup program should be done on the data, since they will treat highly variable data much more carefully than a single pass and prevent it from being flagged as error.

3.2.2 Parameters Used in the Error Detection

"Maximum_smooth_jump" parameter defines how large the first difference between two points can be before the program assumes that there is an error. This parameter is used rarely–only when there are not more than two points available for the

smoothness test. "Smoothness_criteria" is the parameter used in the smoothness test. The larger this parameter is the more variable the valid data is allowed to be by the program. For the D1 data "maximum_smooth_jump" parameters were set to 15 and "smoothness_criteria" parameters were set to 16 for channels 1,2, and 3. "Maximum_smooth_jump" parameters were set to 60 and "smoothness_criteria" parameters were set to 400 for channels 5,7, and 8. "Light_limit" and "tight_lmit" specify how close the data values have to be to their mean when determining the end of error interval. "Light_limit" was set to equal "maximum_smooth_jump" for all channels, while "tight_limit" was equal to just 1/3 of it.

3.2.3 Error Identification

In order to identify the type of error, five different parameters for each error are found. Then, the error type is simply assigned on the basis of these parameters. These parameters are: entrance, exit, length, variability, and error flagging.

Entrance and exit

Both "entrance" and "exit" can have one of three different values—smooth, abrupt, and unknown. "Entrance" is considered smooth if there are four data points with error bit cleared immediately before the error and they, together with the first erroneous value, pass the smoothness test. If they don't pass the test, entrance is considered "abrupt". If there are not four available points (because some of them are marked as errors or are part of the time code) then the entrance is "unknown".

Length

The length is not used literally but is divided into several groups—less than 21, between 21 and 100, between 101 and 200, and larger than 200.

Variability

There are many different variabilities that an error can get assigned: smooth, constant, constant except for the last value, entered and/or exited smoothly but jumps to a constant in between, entered and/or exited smoothly but becomes wild in between, jumps to values that are a multiple of 2^n of the correct value, constant at 1023, constant last portion switching to 1023, abnormally constant, and unknown. (Abnormally constant is not used in the present version, but it would be easy to add it since the "hooks" are there.) This parameter is determined by a large if-then structure which simply sequentially tests every case to see if the error matches it.

Error Flagging

This parameter may have one of three values: marked, unmarked, and initial value is unmarked.

These five parameters completely identify the type of error.

3.2.4 How Error Extent is Determined

The program does not simply use the error flag as an indicator of error. There are several checks to ensure that unmarked portions of erroneous data are detected. After the program decides that the error has started it will not assume that it has ended until all of the following are satisfied: the error flag indicates no error, the value came close to the data average in the corresponding channel and the data following the error appears valid.

Here is how it works in detail: first, the program checks sequential data values until it finds data where the error flag is clear. The program then remembers that point. Then it checks if the current value is close enough to the data average over the last 100

valid data points. There are two parameters for this: "light_limit" and "tight_limit". If the value is within "tight_limit" of the average, then the program thinks it is probably the end of the error (but doesn't yet declare it as such). If the value is not within "tight_limit" but is within "light_limit" of the average the program assumes nothing at this point. If the value is outside even "light_limit" then the program assumes that this in not the end of error. Once the program finds that it may be the end of the error interval it checks if the values after the error are smooth. If they are not, the program assumes that this is not the end of error. If they are, it checks if the number of points from the last abrupt value is larger than 10. If it is, then this is the end of the error. Notice that the program keeps checking this for every value until the end of the error is found. These checks are performed for at least five values, but may be performed for much longer if the data stream is not smooth or does not come to within "tight_limit" of the average.

The program also continuously checks data for smoothness even when not within erroneous data. This way the program will not miss errors that either start unmarked or are entirely unmarked. This is the only test used to find completely unflagged errors.

3.2.5 Error Handling

Once the error type is identified the error may be cleared—i.e. the error flag will be set to 0. It may be set—i.e. error flag will be set to 1. The data may be shifted by a power of 2 and the error flag cleared—its values will be divided or multiplied by some 2^n and the error flag set to 0. The error may be "corrected"—its value set to some number, Lastly it may be left alone, and just reported. The type of action can be set independently for each type of error by changing the parameters at the end

of ERROR_TYPE subroutine. For this analysis the data was never shifted if the n required in 2^n was greater than 2, also no data values were changed other than by shifting.

Chapter 4

Results

4.1 Data Analysis

Statistical methods were used to analyze the data – see the Experimental Methods chapter above. The main emphasis was on determining the relationship between SMS and subject discomfort. The main questions were:

- Is there a statistically significant difference between the levels of astronaut head activity during the periods when they were experiencing SMS and when they were not? A t^2 parameter was computed, and Chebyshev's inequality used to determine statistical significance.
- Is there a relationship between SMS susceptibility and natural head activity level when asymptomatic?
- If yes, could the neck collar be used to reduce the head motion (and perhaps the incidence of SMS?)

This chapter presents the results of the data analysis. First, the error detection is discussed. Then the motion indices plots, both detailed and ARU session means, together with the discomfort indices are presented. Then a comparison between the subjects is done. Their SMS ranking and unconstrained head motion levels when

asymptomatic are compared. And last, the cross plots of the motion indices versus discomfort indices are presented.

Discomfort index data for D1 mission was computed from several sources—mainly from the transcripts of the in-flight voice recorder tapes that the astronauts used to record their symptoms. In addition, plots of discomfort vs. MET obtained in post flight debriefing were also used. SL1 discomfort vs. MET was taken from McCoy's thesis. The discomfort indices were compiled from all sources, with the best estimate taken where the different reports may be slightly conflicting.

4.2 Results of Error Detection

Extensive manual examination of the cleaned D1 data confirmed the effectiveness of the ERROR_CLEANUP program. To verify that the unmarked errors had indeed been detected values from one of the D1 data files were examined completely point by point. Large portions of the other six files were examined as well (given the typical 45 MB size of these files it was not feasible to manually examine all of them in their entirety). In all files the motion index statistics have been reduced to more reasonable levels. All data where the error rate or the activity indices differed greatly before and after error processing by the Error_cleanup program were also examined manually. As a result, when the activity data was compared to the mission logs very good agreement was found. For example – when ARU was not worn the motion index dropped to negligible levels. It was also possible to identify in time specific activities–hop and drop, dome experiment, neck collar experiment – simply by looking at ARU data. In one case it was suspected that subject I wore the neck collar for only 30 minutes of the 3 hour scheduled period. This was later confirmed when subject I consulted his mission notes.

Due to time constraints, extensive manual examination of Spacelab 1 data was

not performed. SL1 analysis relied exclusively on the playback unit and the ERROR_CLEANUP program to detect the errors. However, only the angular channels were used in the SL1 analysis, and the accelerometer gains on SL1 were identical to D1. Therefore, the program was expected to detect errors in SL1 data as well as it did in D1 data.

4.3 Results of Statistical Analysis - Motion and Discomfort Indices Plots

There was a difference in the head motion between the periods for the subject I when symptomatic vs. asymptomatic. This difference was statistically significant ($p < .05$) by Chebyshev for all ARU sessions. See Figures 4.2 and 4.3 for t^2 parameters. There was no such difference for the subject J who experience only minor SMS symptoms. Also, the average level of activity for this subject was lower.

ARU session time MET	12-20	43-50	104-113	130-133
12-20	-	35.48	26.55	22.49
43-50	35.48	-	.276	.25
104-113	26.55	.276	-	.01
130-133	22.49	.25	.01	-

Figure 4.1: Values of t^2 for the difference between ARU sessions for subject I

ARU session time MET	4-8	24-29	79-83	105-107
4-8	-	.549	1.57	.05
24-29	.549	-	1.75	.072
79-83	1.57	1.75	-	.024
105-107	.05	.072	.024	-

Figure 4.2: Values of t^2 for the difference between ARU sessions for subject J

Likewise, there was statistically significant difference between the motion indices for subject B on SL1 during vs after sickness. Since there were many more ARU sessions for this subject than for subject I, the t^2 parameter was computed for all the data

while symptomatic vs. all the data when asymptomatic[1]. The t^2 was found to be 87.63, which gives $p < .0114$ by Chebyshev's ineqality.

However, there was no similar difference for subject C, who did experience SMS. A possible explanaition lies in the character of his symptoms. Subject C, unlike subjects B and I, did not experience a sustained increased discomfort during the first few days. Instead he experienced several sudden episodes of severe discomfort (vomiting) but felt relatively well the rest of the time. Perhaps, because he felt better, on average, than subject B, he was less compelled to limit his head movements. It is interesting to note that McCoy concluded that subject C's activity had increased late in the mission. However, his conclusion was biased by the linear channels, and also by a single high activity session at the end of the mission. It is not so clear whether the motion index during the last session of mission represents a trend toward increasing activity or not. Based on the consistency of other points, we believe this is not the case.

[1]Files S2, S32, S3 were used for the symptomatic period, and files S9, S8, S14, S16, and S17 were used for the asymptomatic period. File S4 was not used due to high error rate, while file S13 was not used because it was suspected that ARU was not worn during this period.

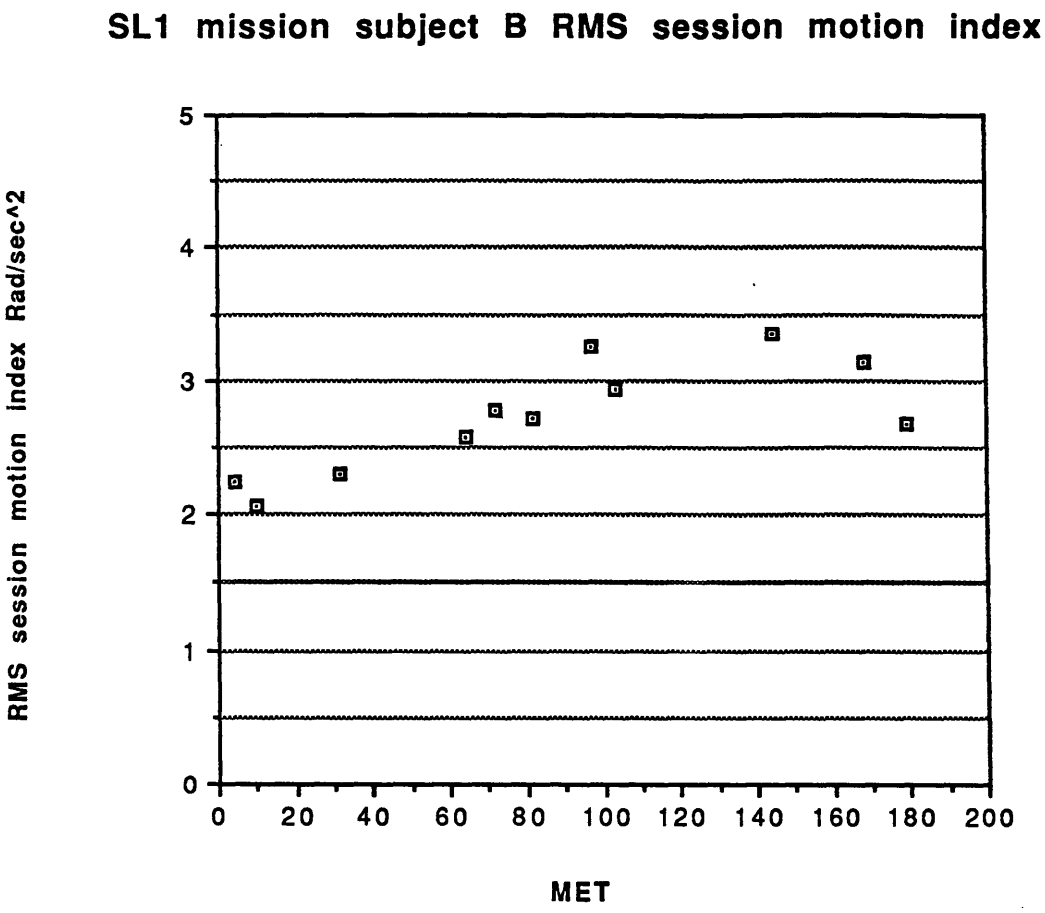

Figure 4.3: Subject B ARU session RMS motion indices

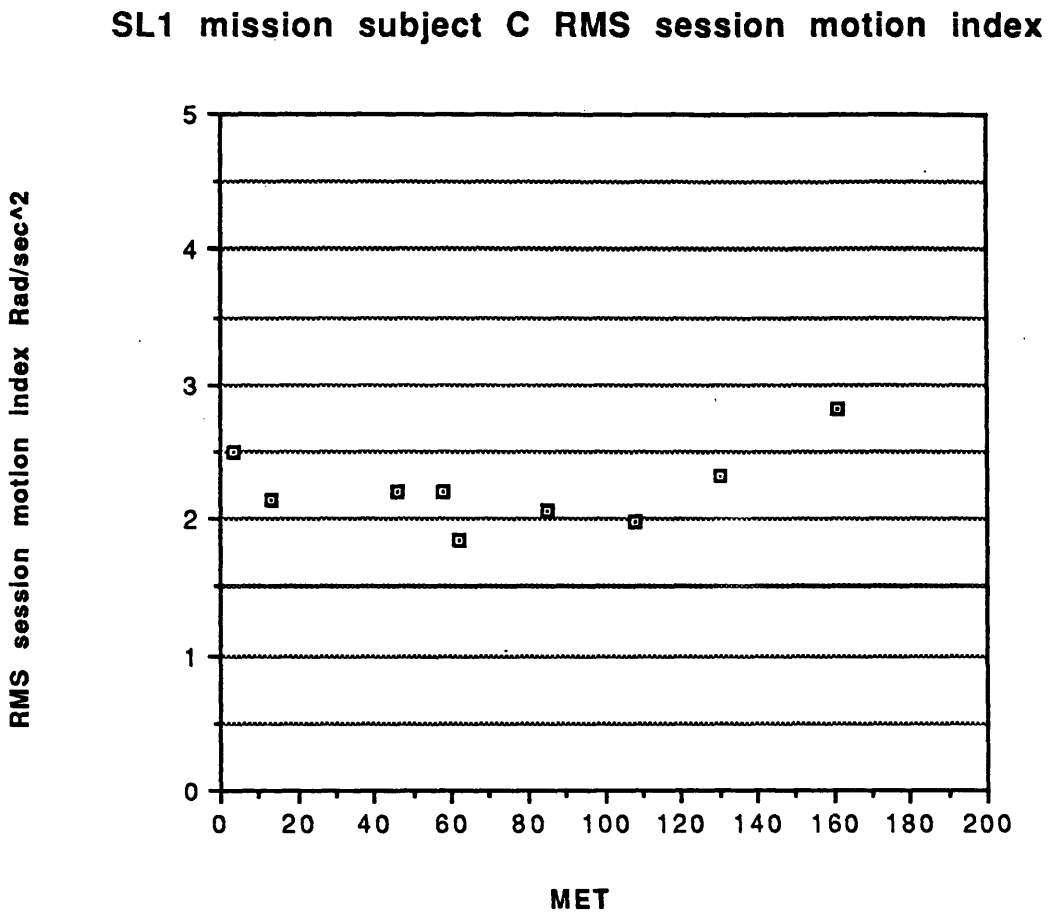

Figure 4.4: Subject C ARU session RMS motion indices

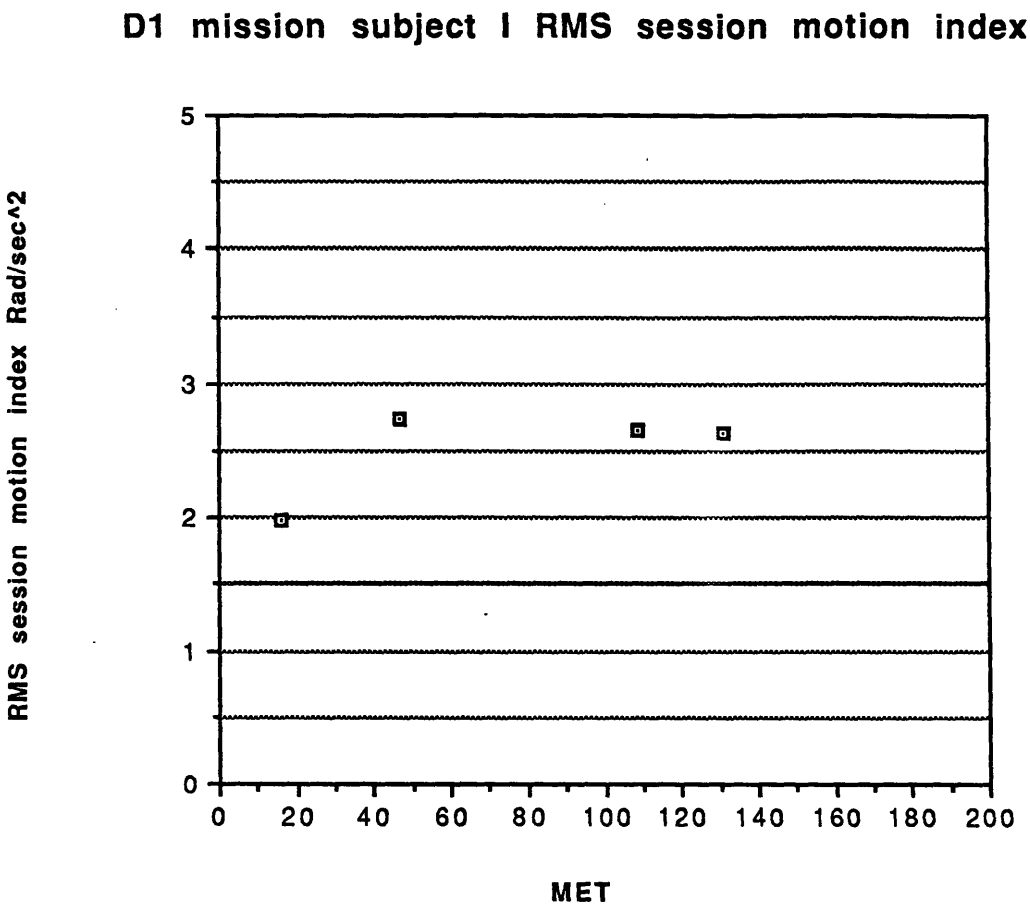

Figure 4.5: Subject I ARU session RMS motion indices

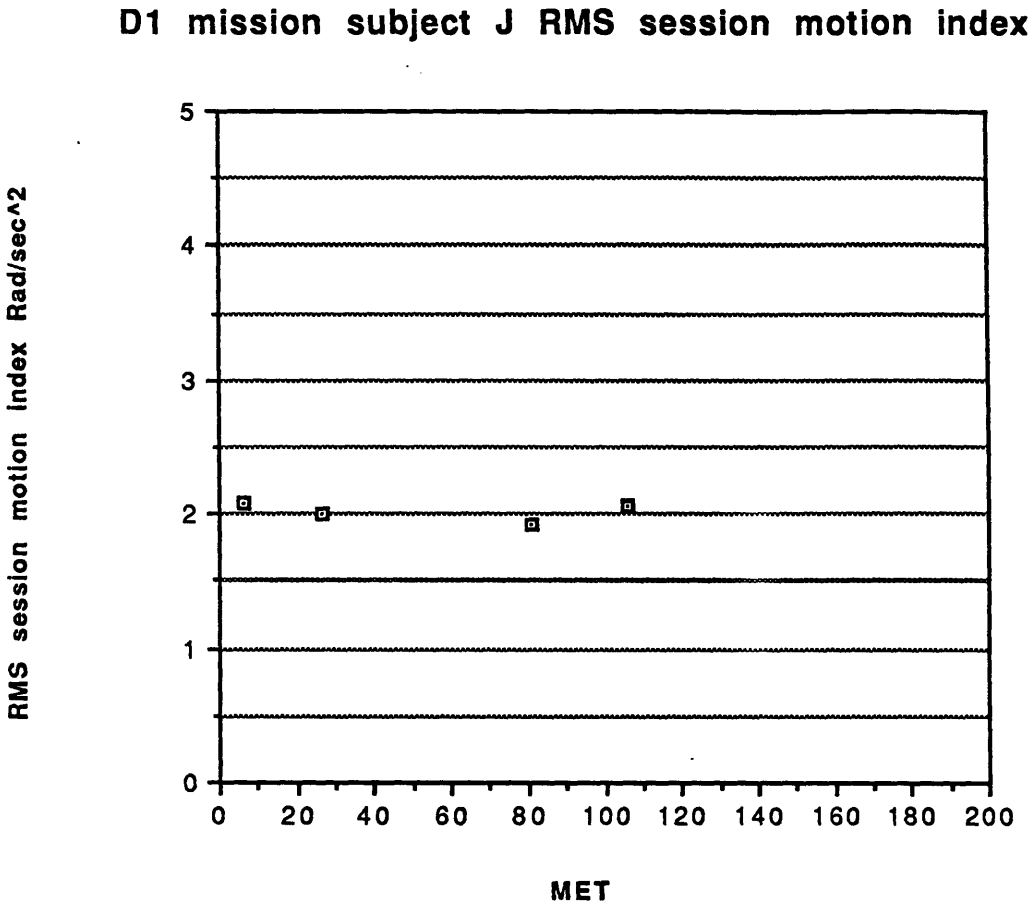

Figure 4.6: Subject J ARU session RMS motion indices

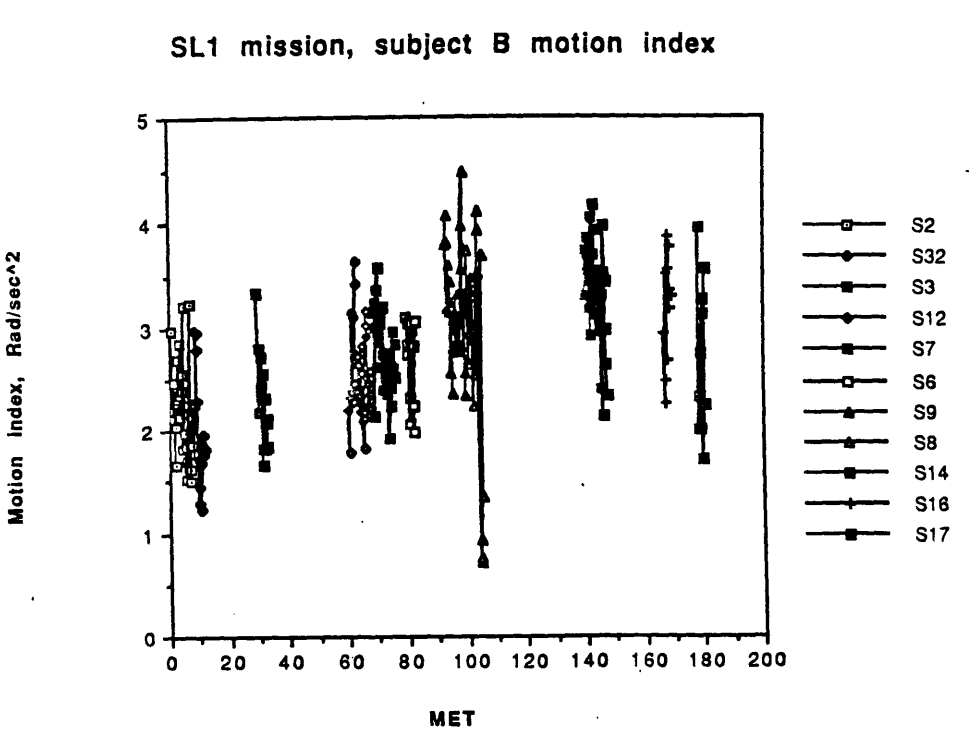

Figure 4.7: Subject B motion index

Below are the detailed motion indices and discomfort plots by ARU session:

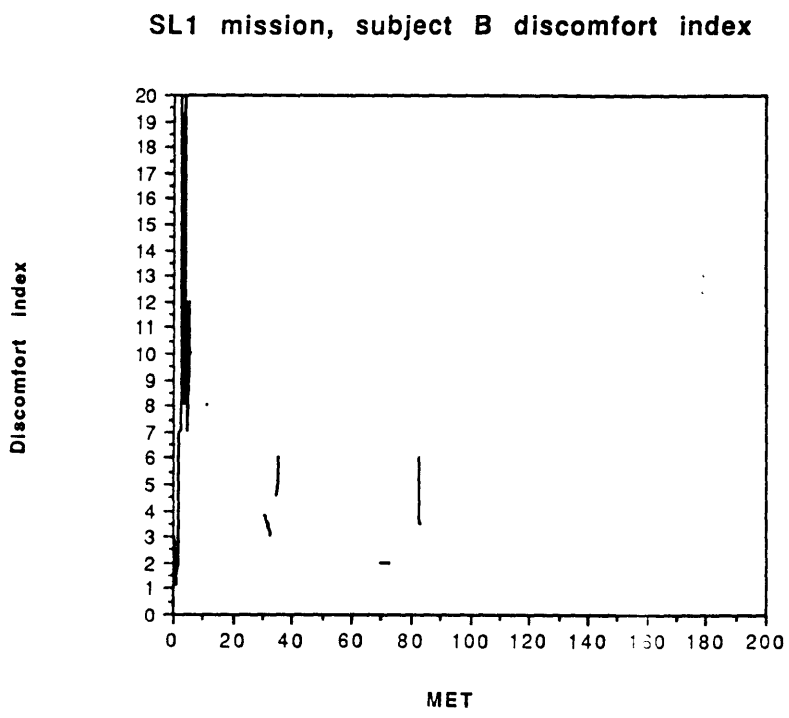

Figure 4.8: Subject B discomfort index

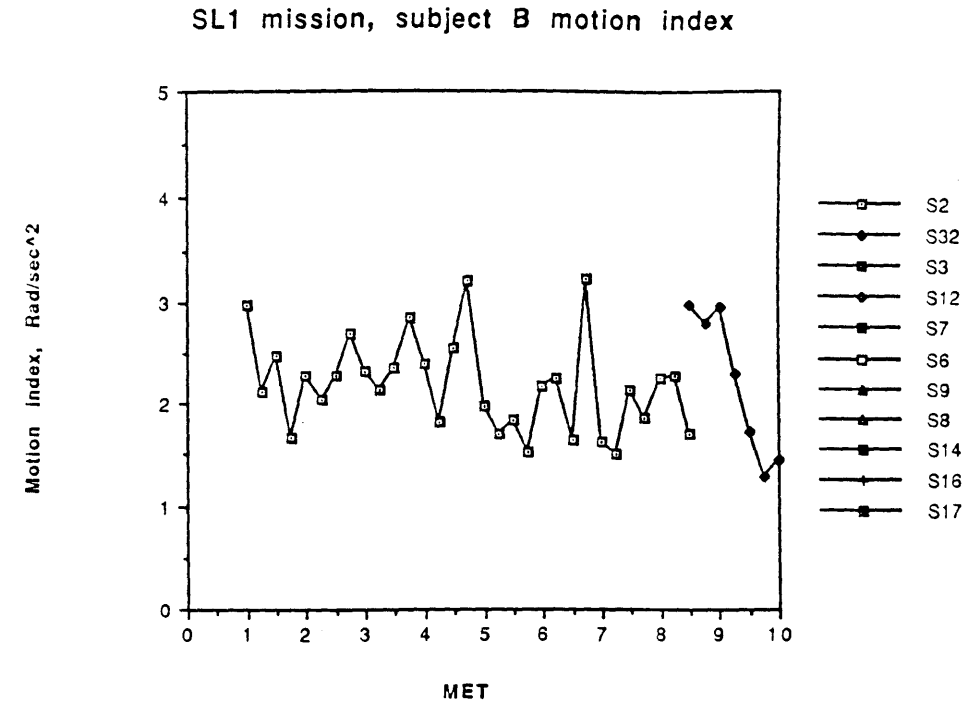

Figure 4.9: Subject B motion index

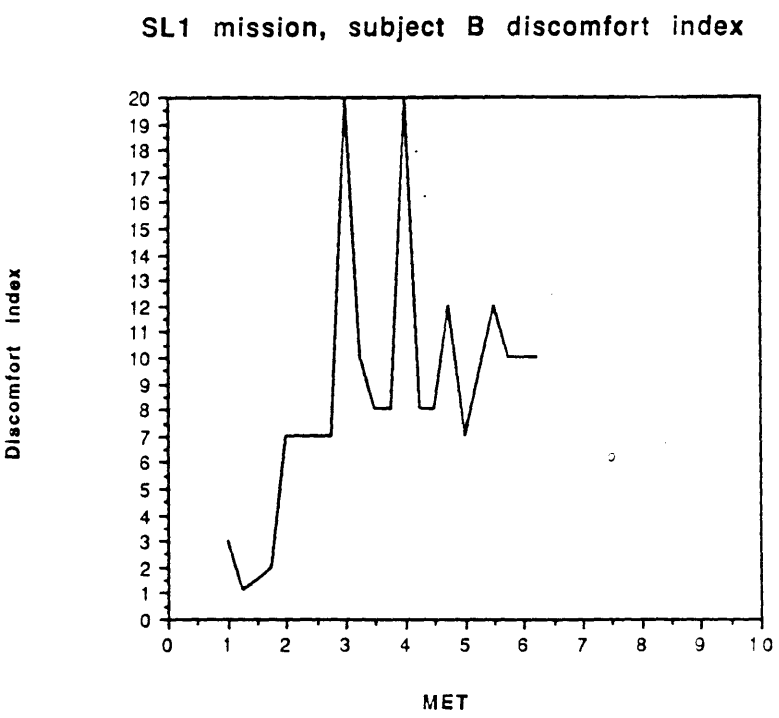

Figure 4.10: Subject B discomfort index

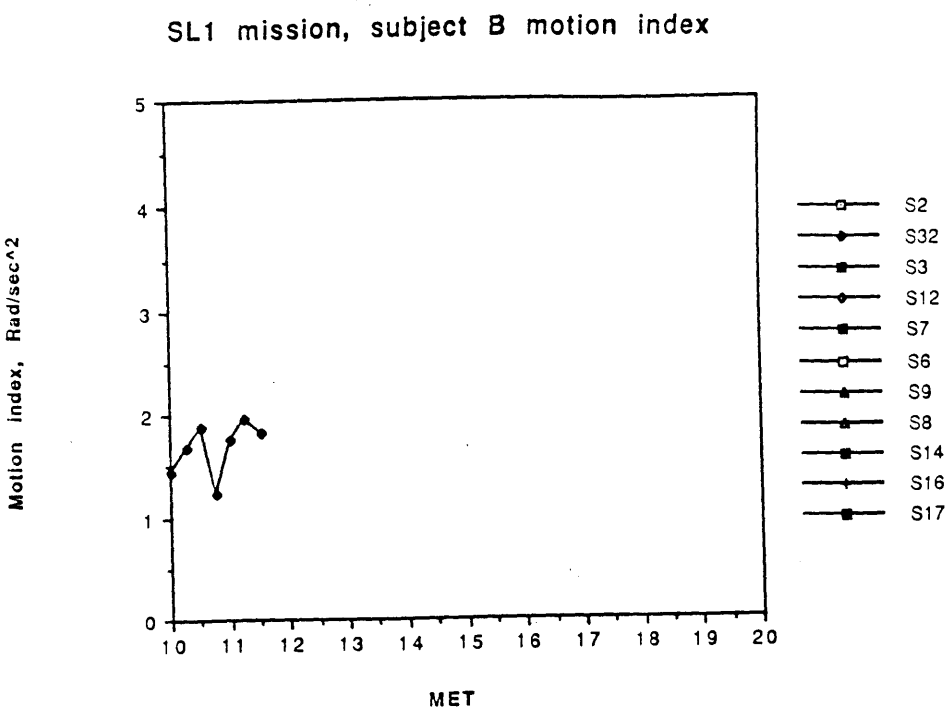

Figure 4.11: Subject B motion index

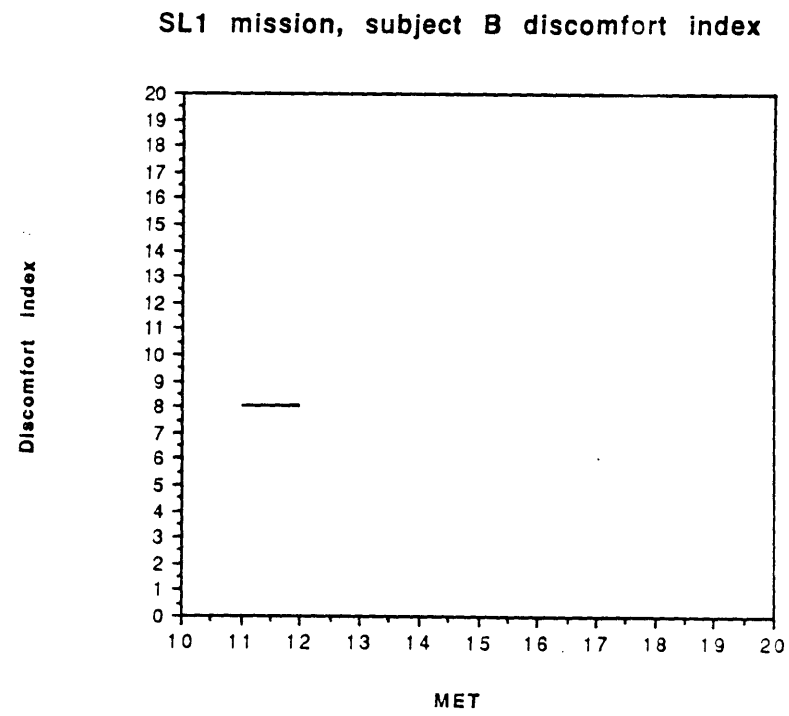

Figure 4.12: Subject B discomfort index

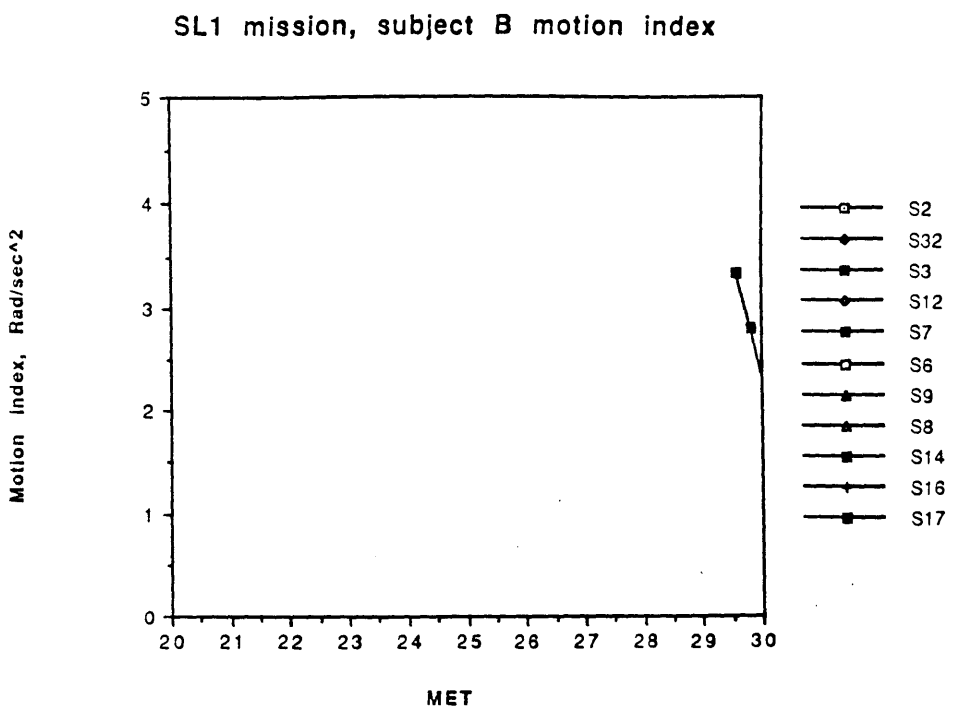

Figure 4.13: Subject B motion index

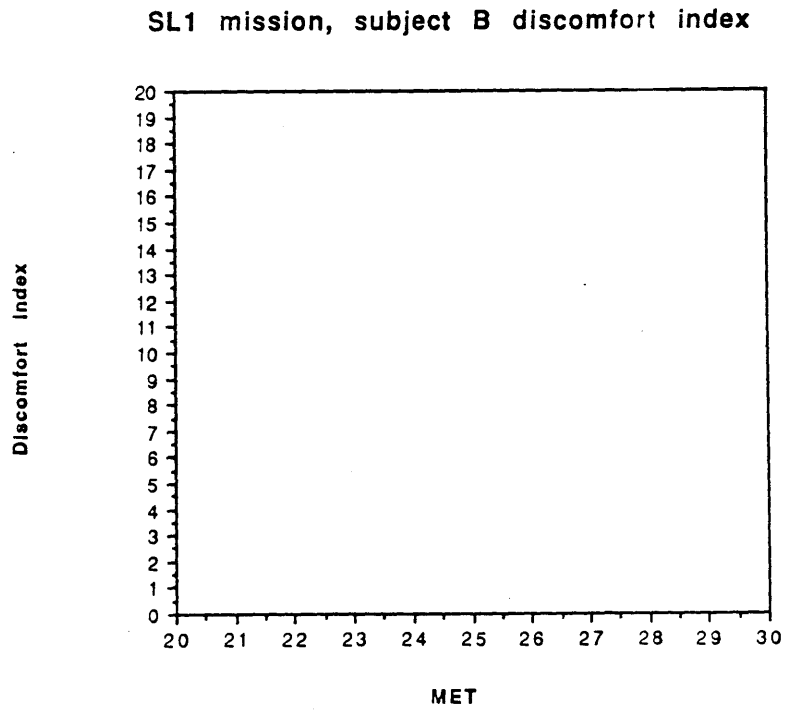

Figure 4.14: Subject B discomfort index

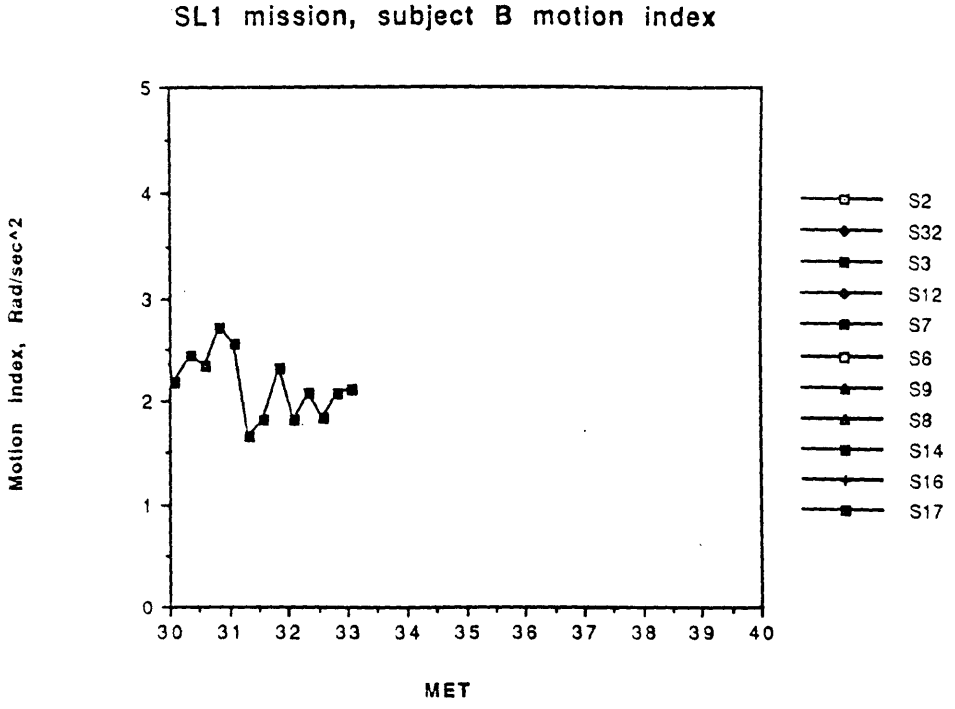

Figure 4.15: Subject B motion index

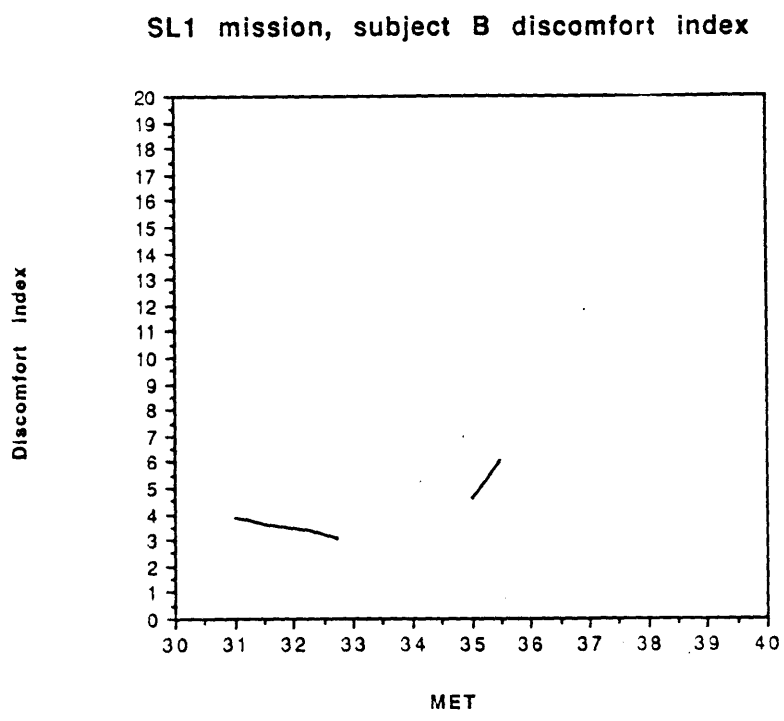

Figure 4.16: Subject B discomfort index

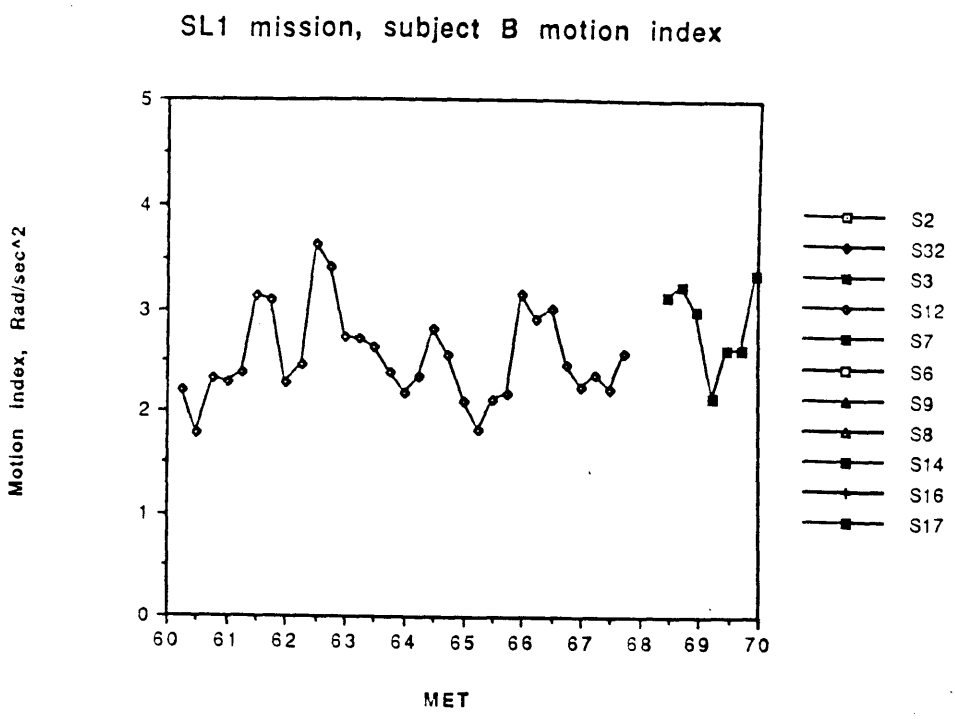

Figure 4.17: Subject B motion index

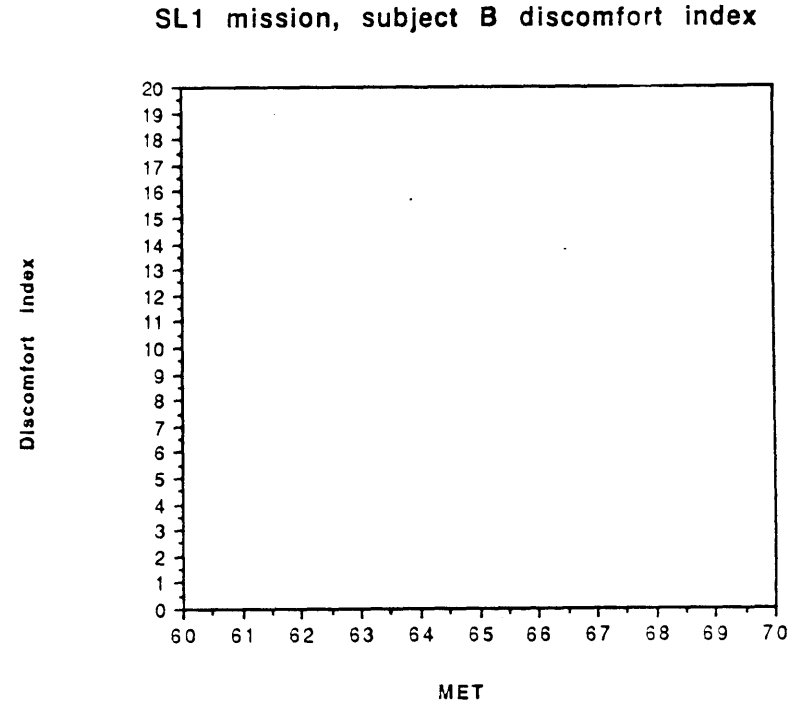

Figure 4.18: Subject B discomfort index

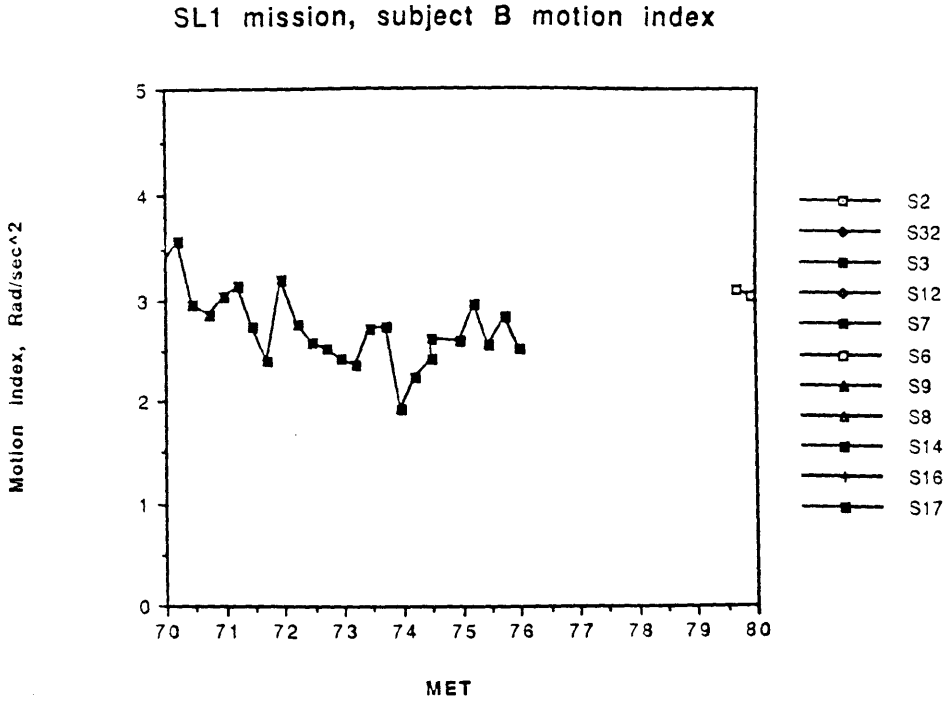

Figure 4.19: Subject B motion index

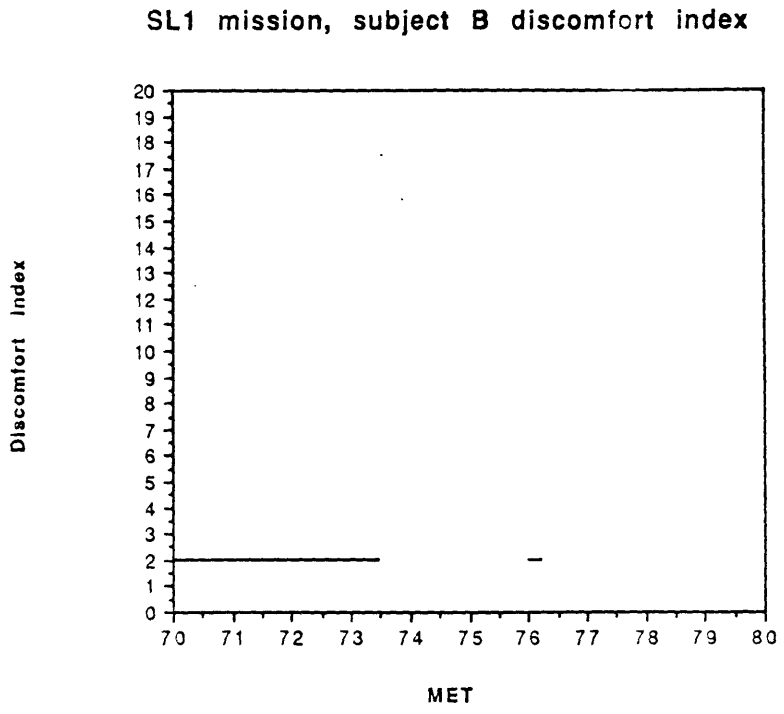

Figure 4.20: Subject B discomfort index

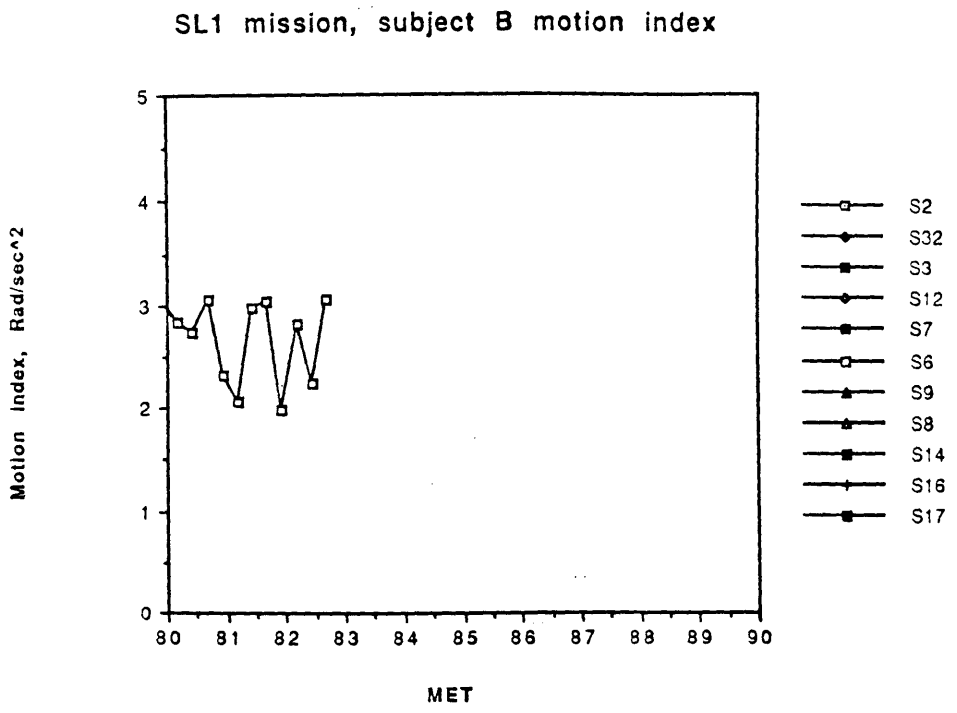

Figure 4.21: Subject B motion index

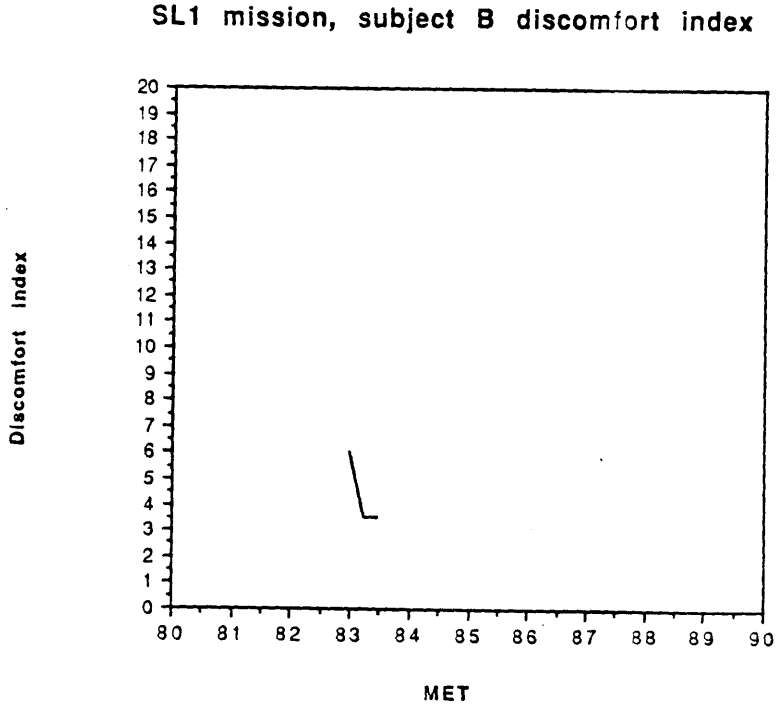

Figure 4.22: Subject B discomfort index

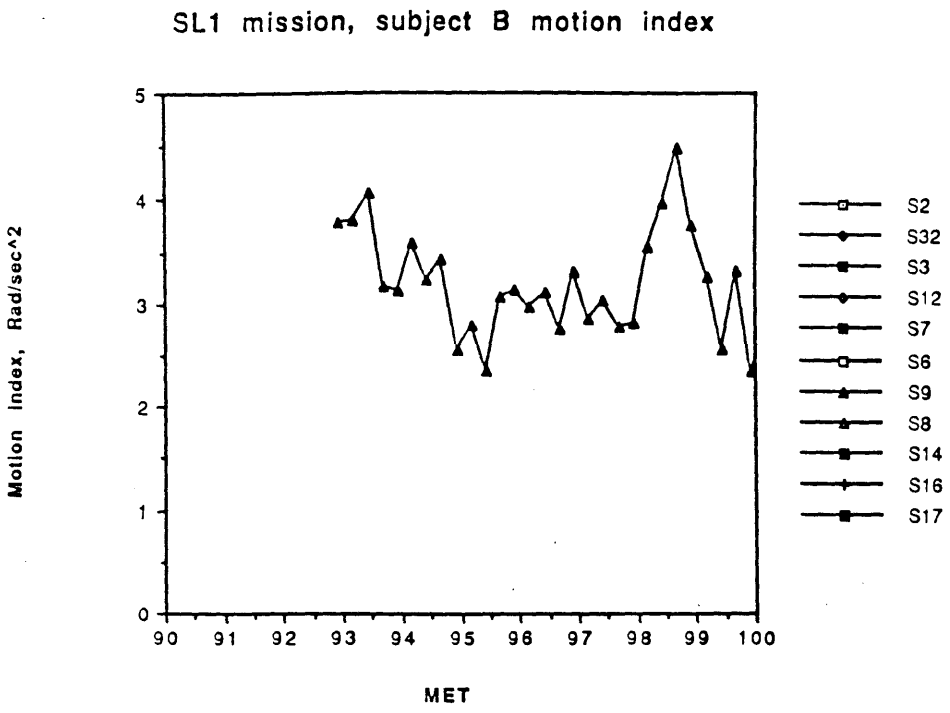

Figure 4.23: Subject B motion index

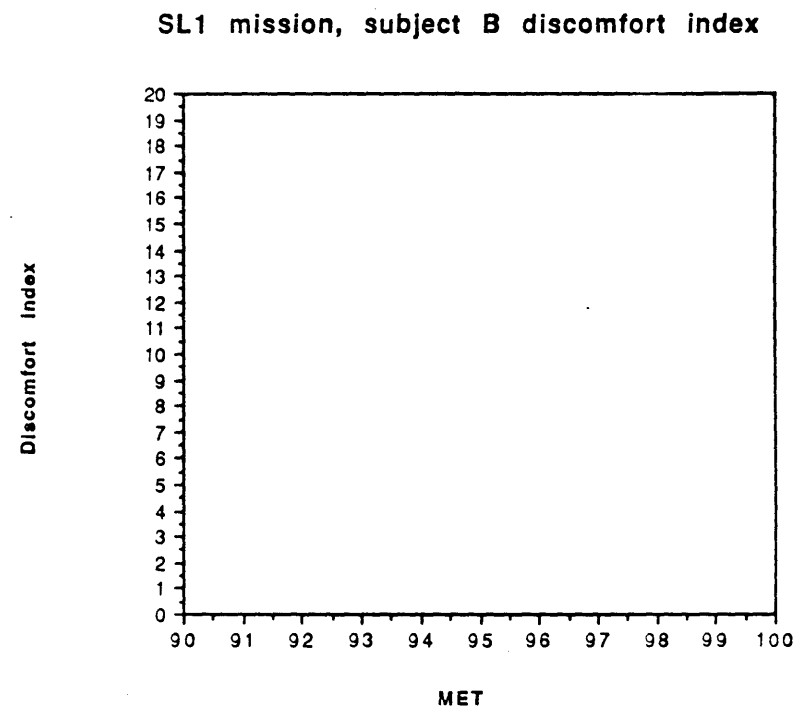

Figure 4.24: Subject B discomfort index

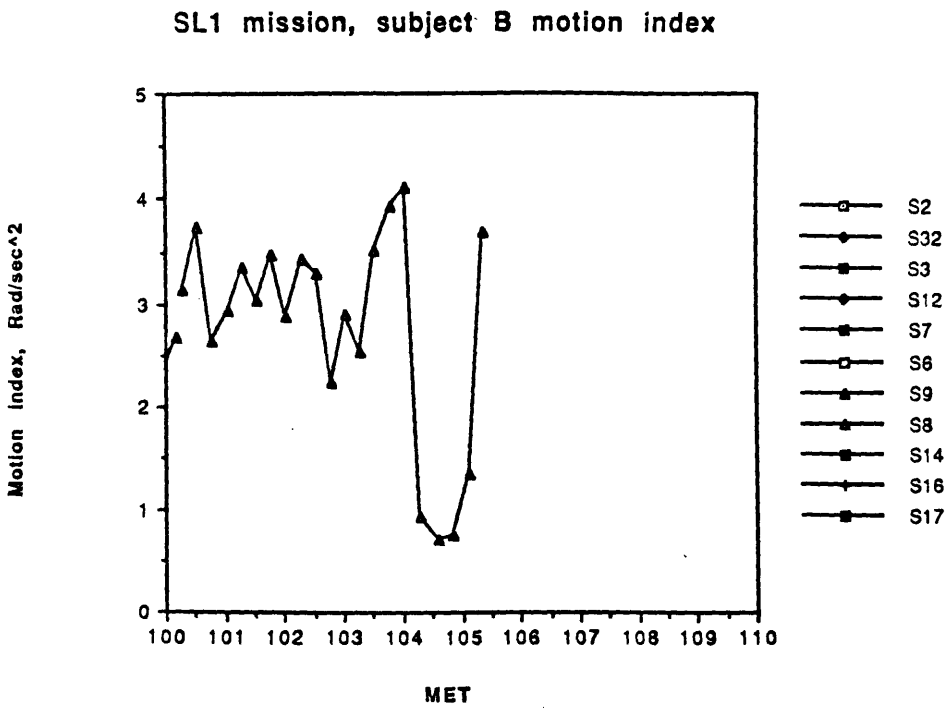

Figure 4.25: Subject B motion index

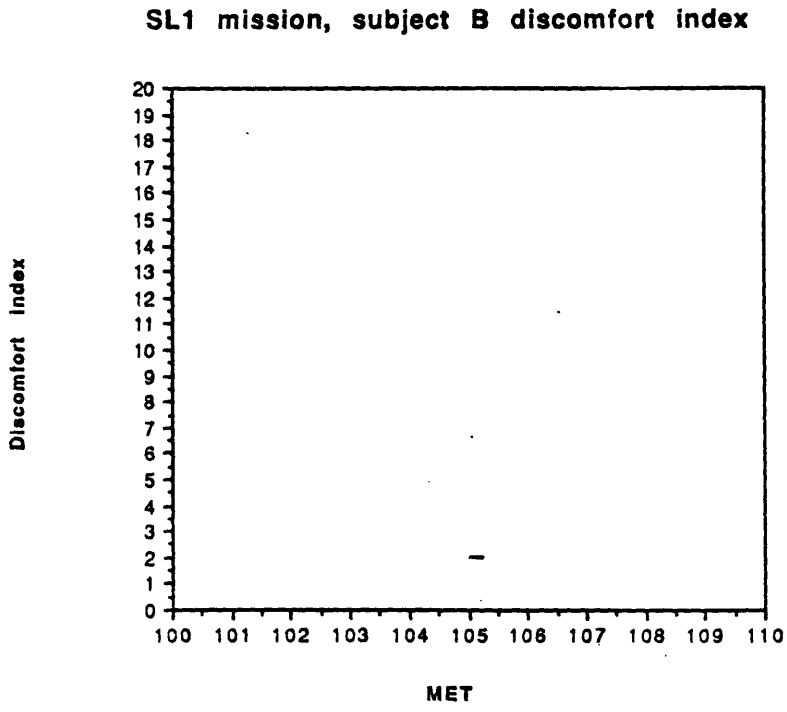

Figure 4.26: Subject B discomfort index

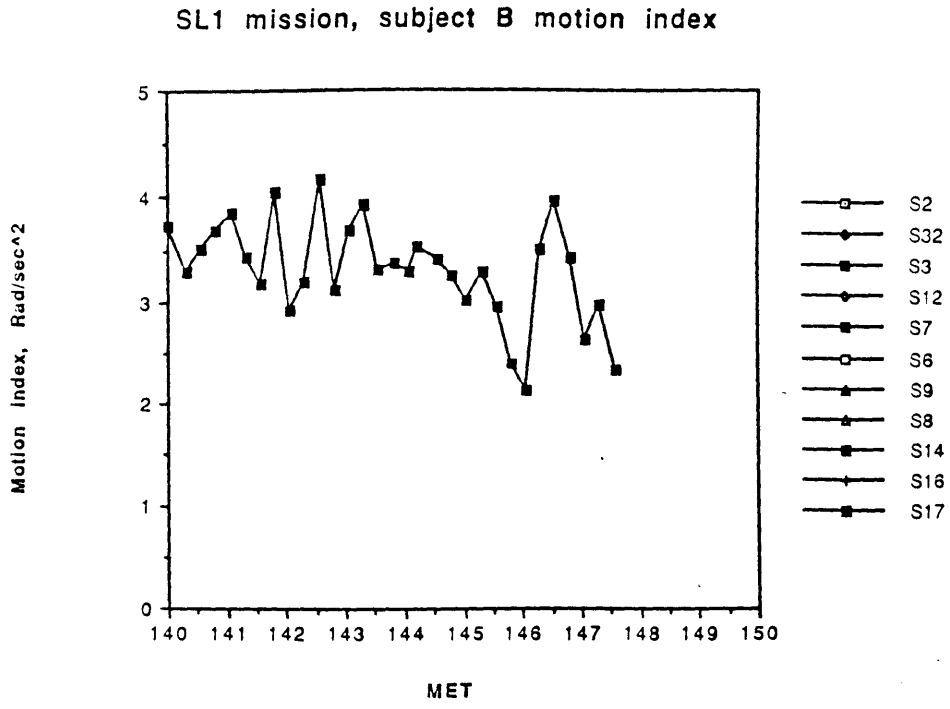

Figure 4.27: Subject B motion index

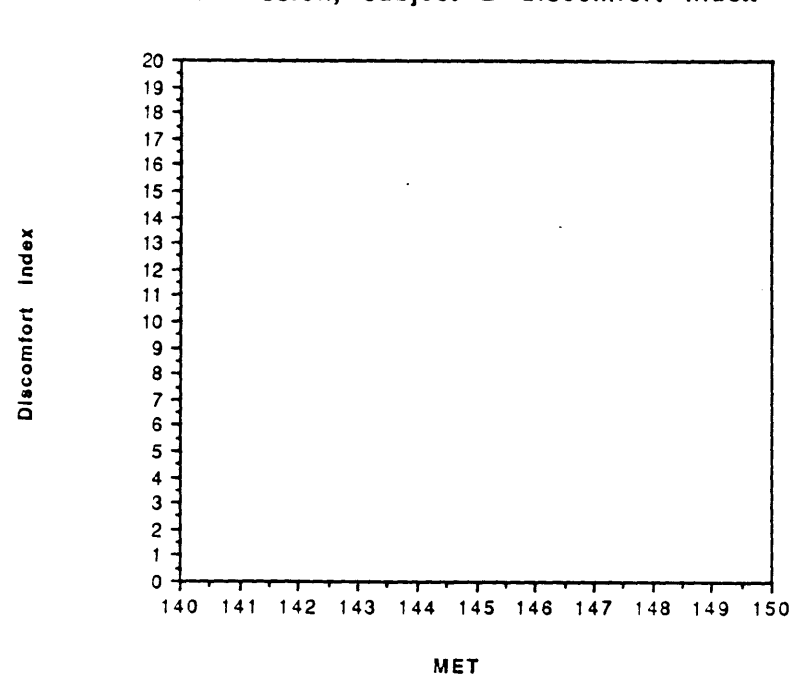

Figure 4.28: Subject B discomfort index

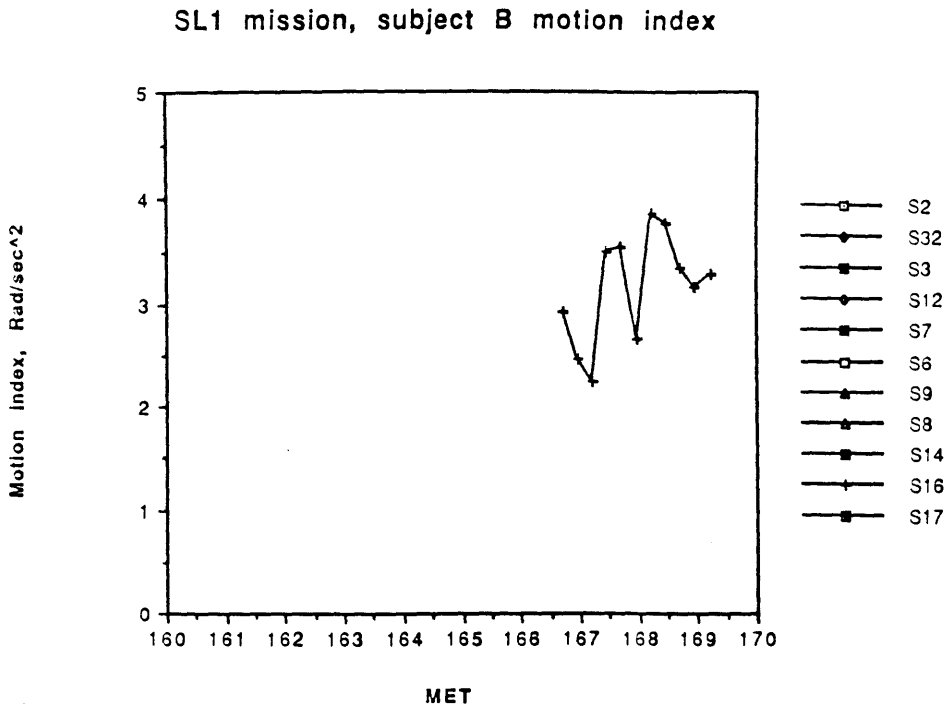

Figure 4.29: Subject B motion index

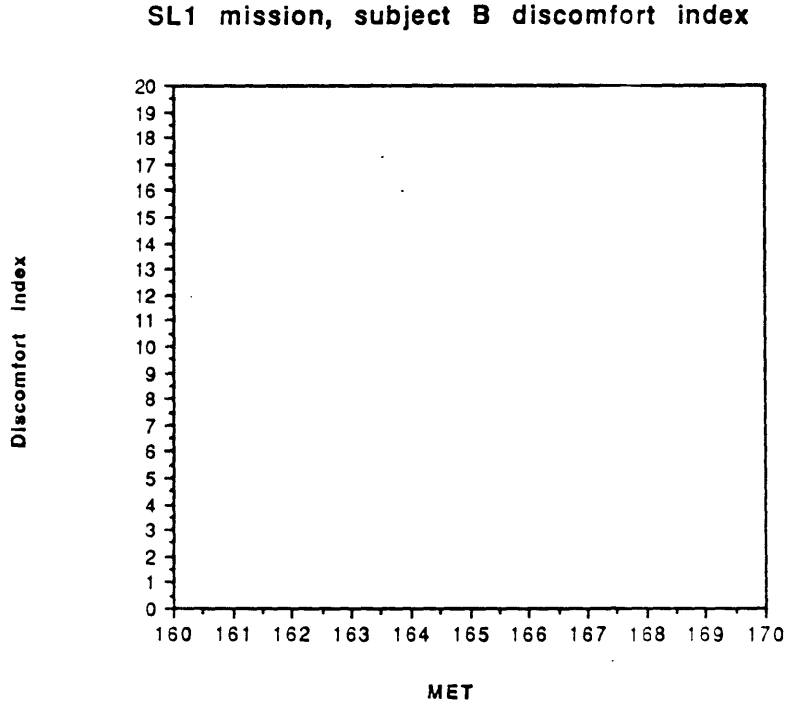

Figure 4.30: Subject B discomfort index

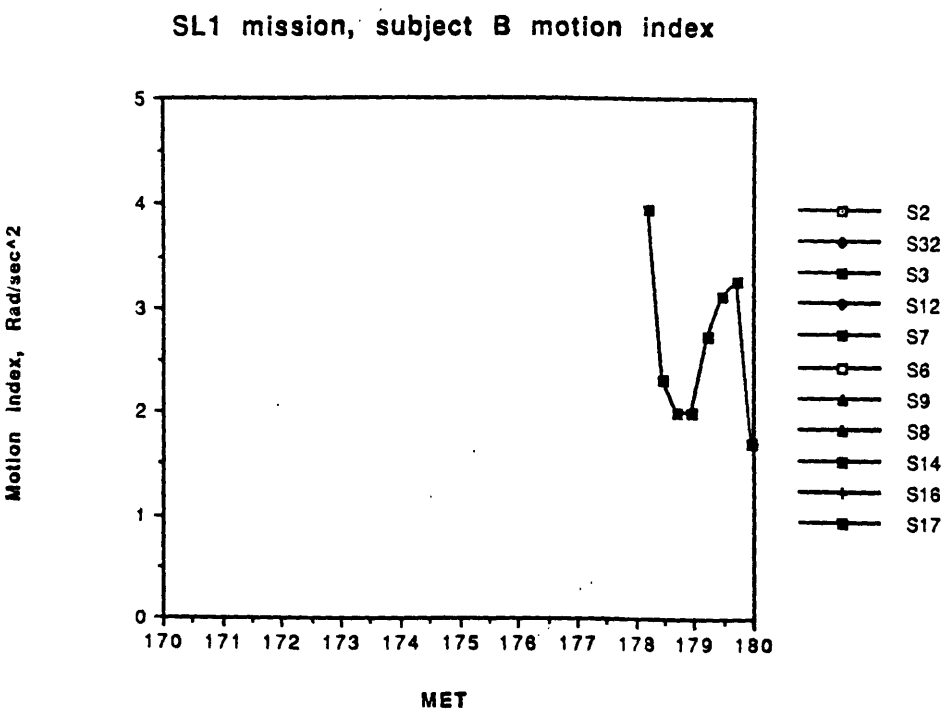

Figure 4.31: Subject B motion index

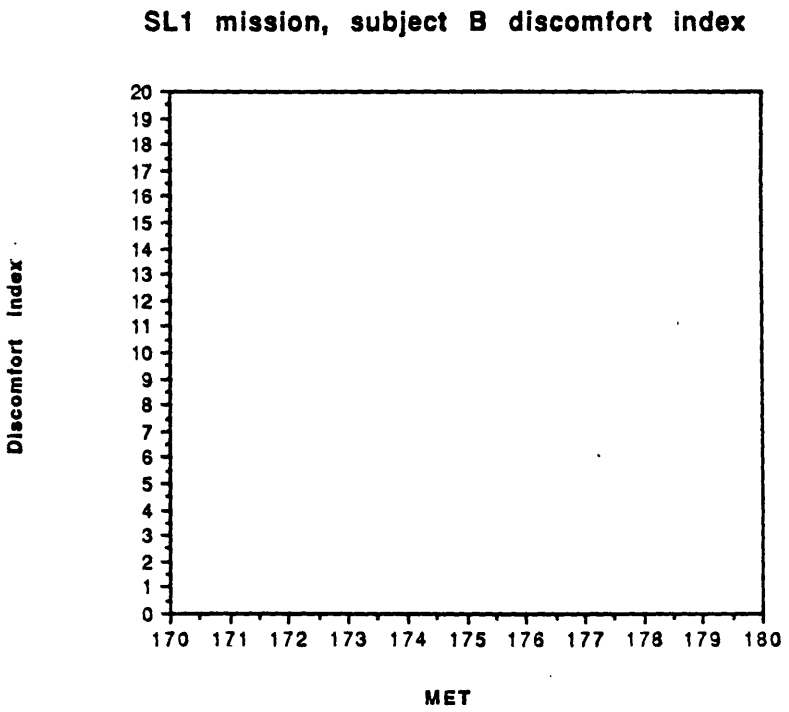

Figure 4.32: Subject B discomfort index

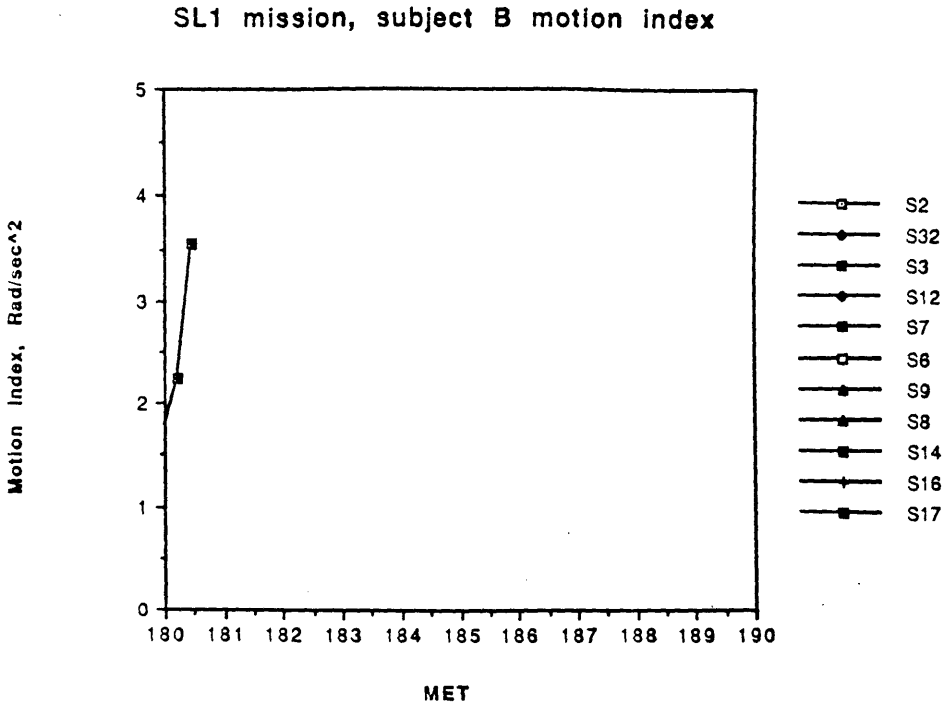

Figure 4.33: Subject B motion index

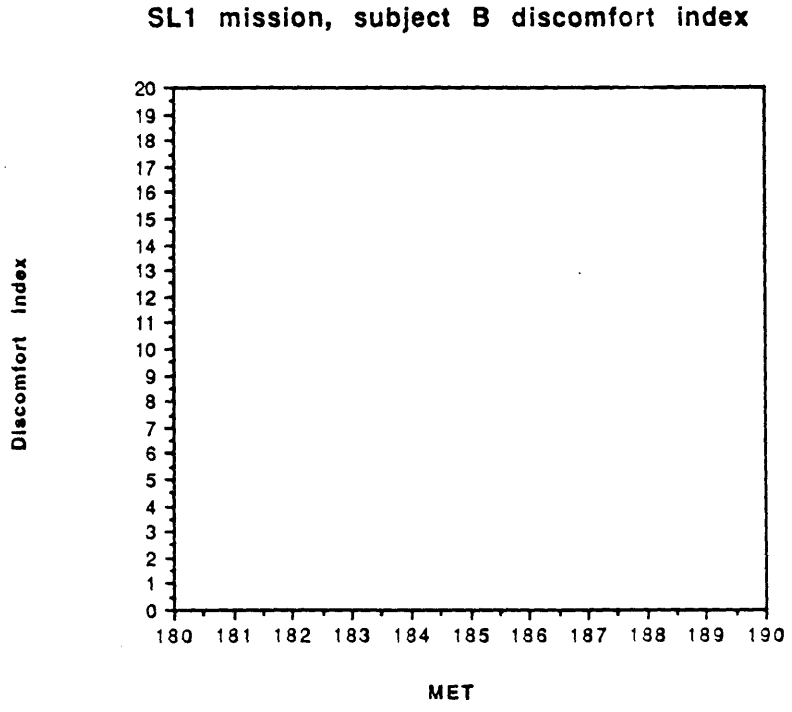

Figure 4.34: Subject B discomfort index

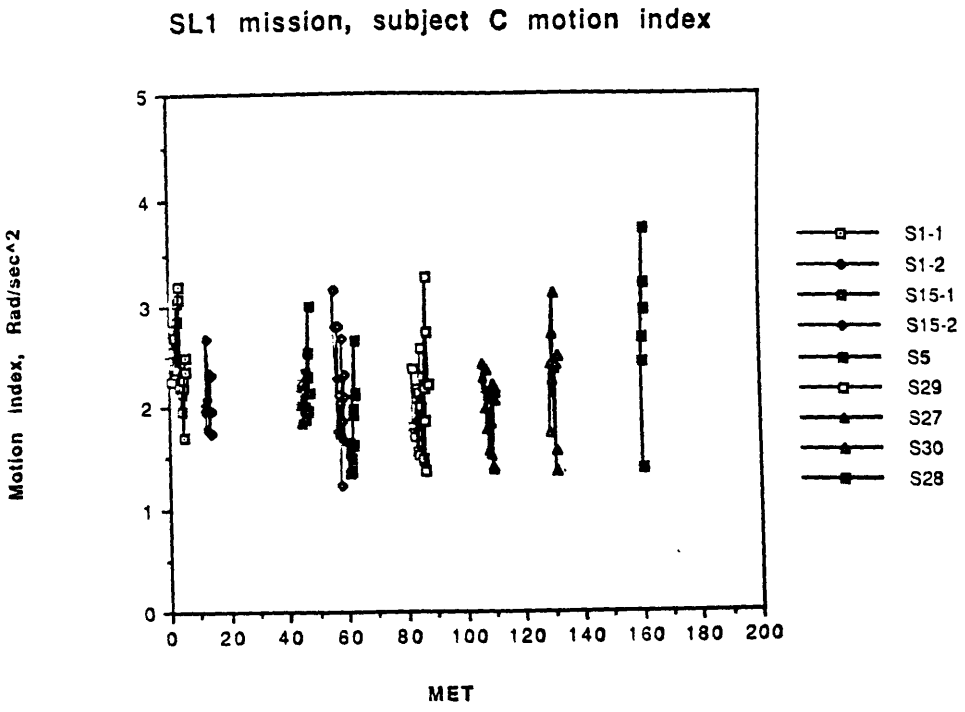

Figure 4.35: Subject C motion index

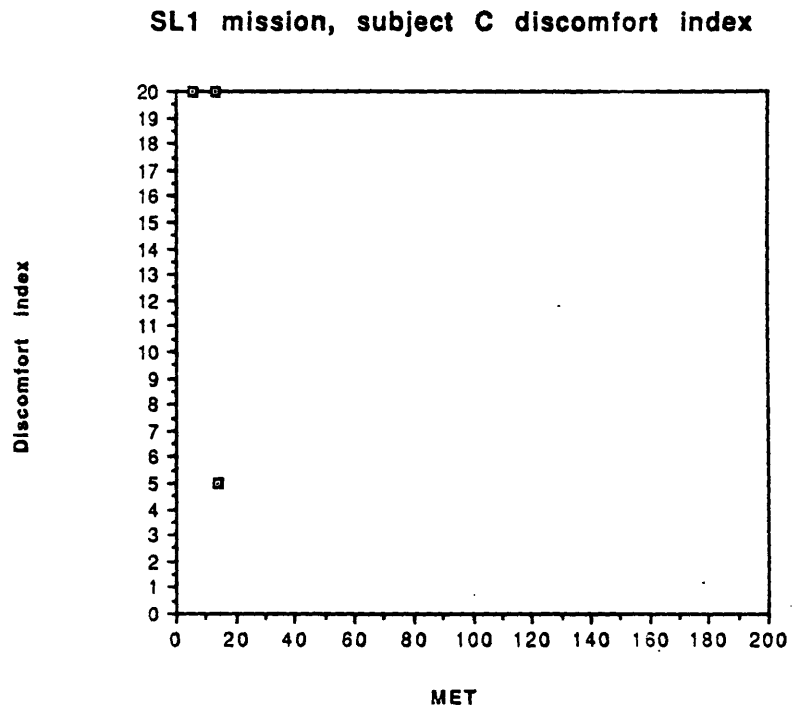

Figure 4.36: Subject C discomfort index

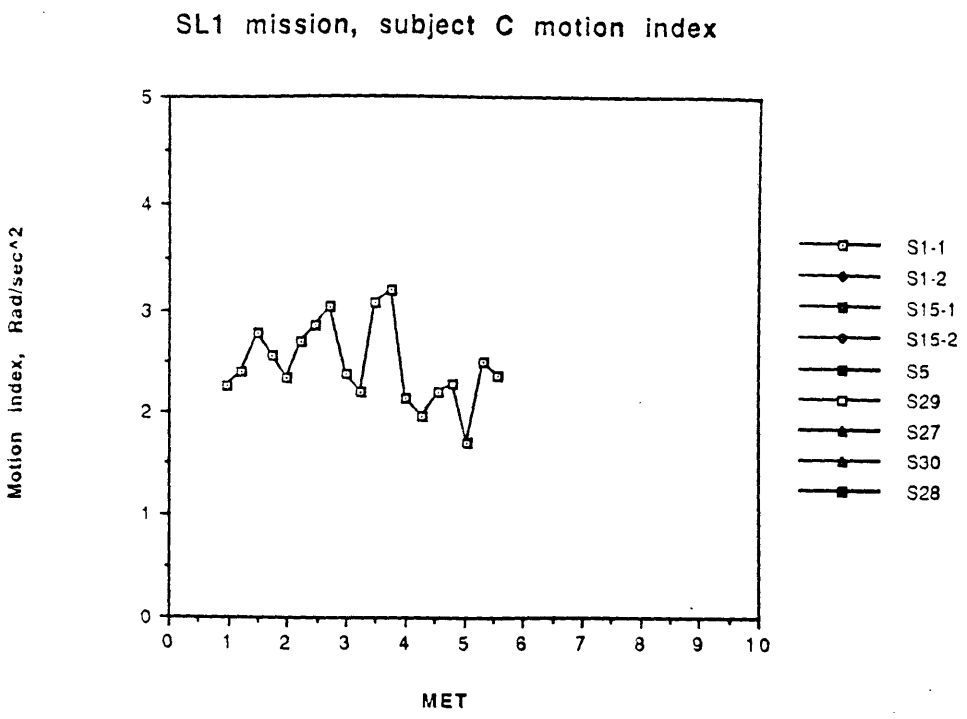

Figure 4.37: Subject C motion index

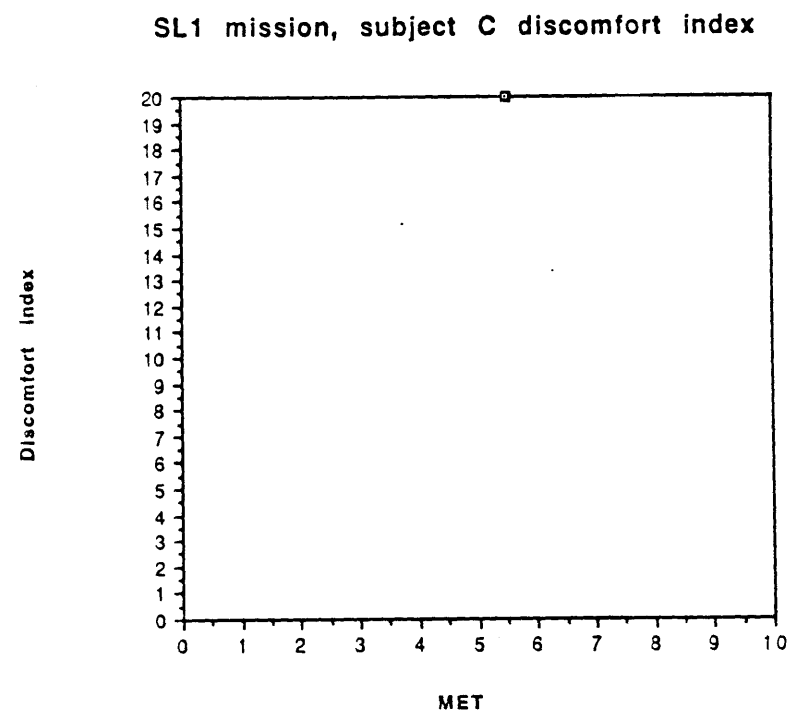

Figure 4.38: Subject C discomfort index

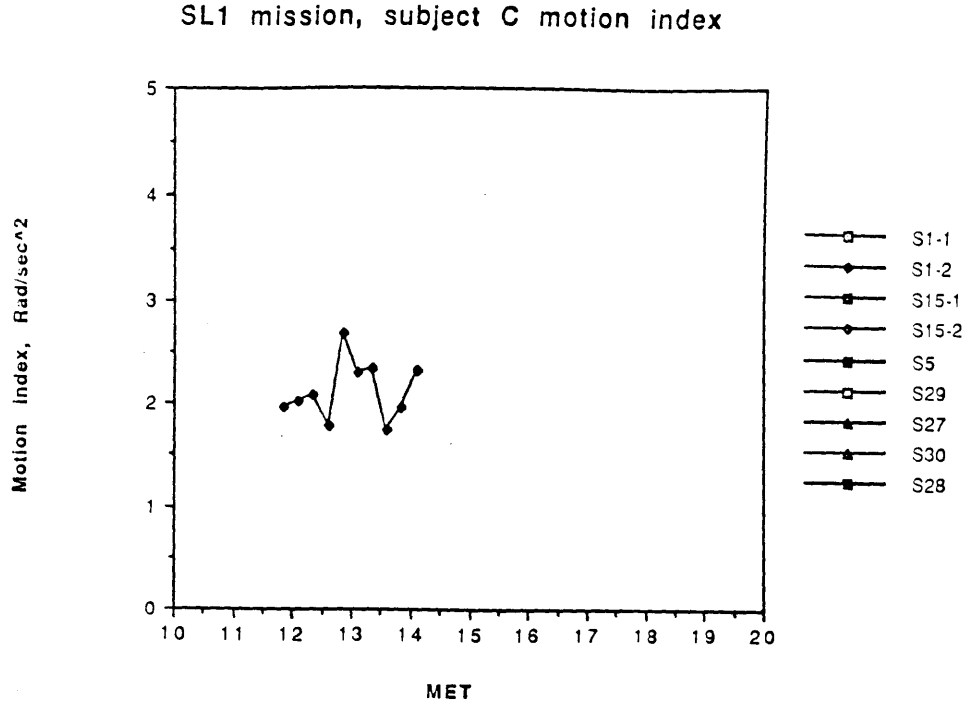

Figure 4.39: Subject C motion index

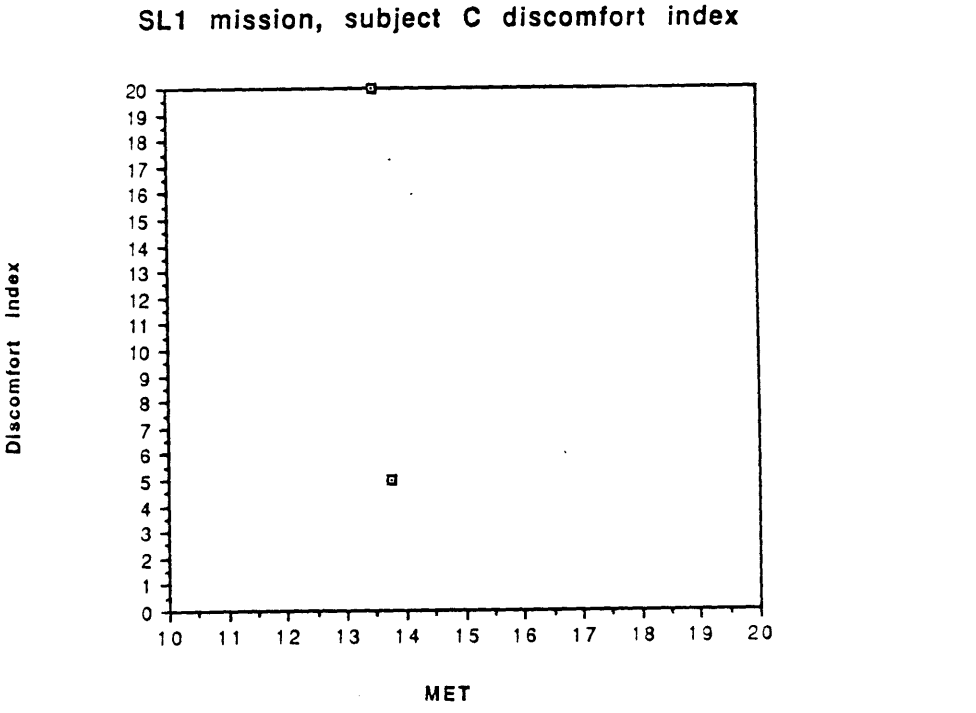

Figure 4.40: Subject C discomfort index

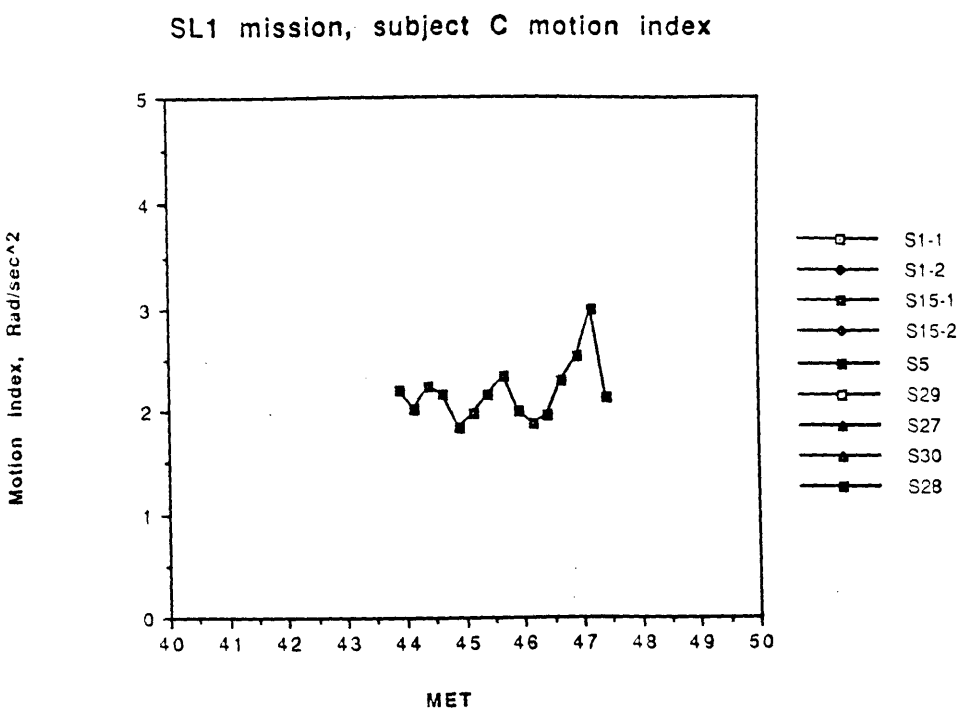

Figure 4.41: Subject C motion index

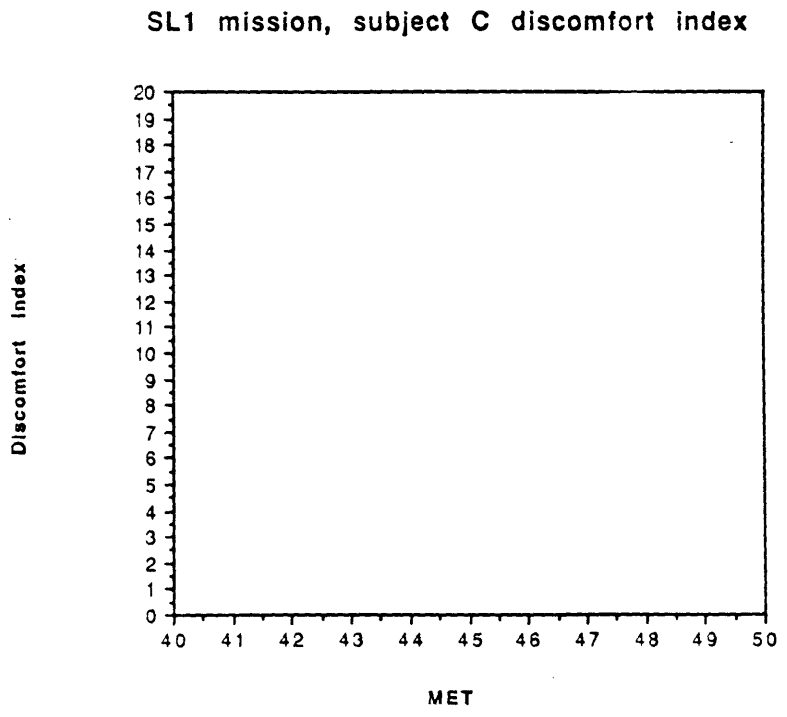

Figure 4.42: Subject C discomfort index

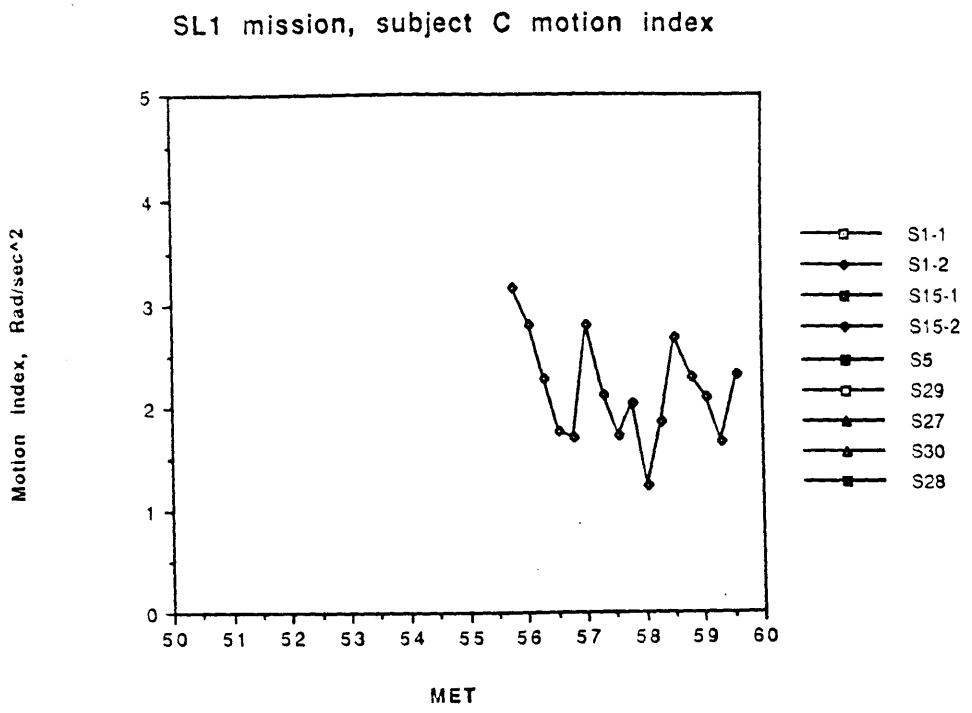

Figure 4.43: Subject C motion index

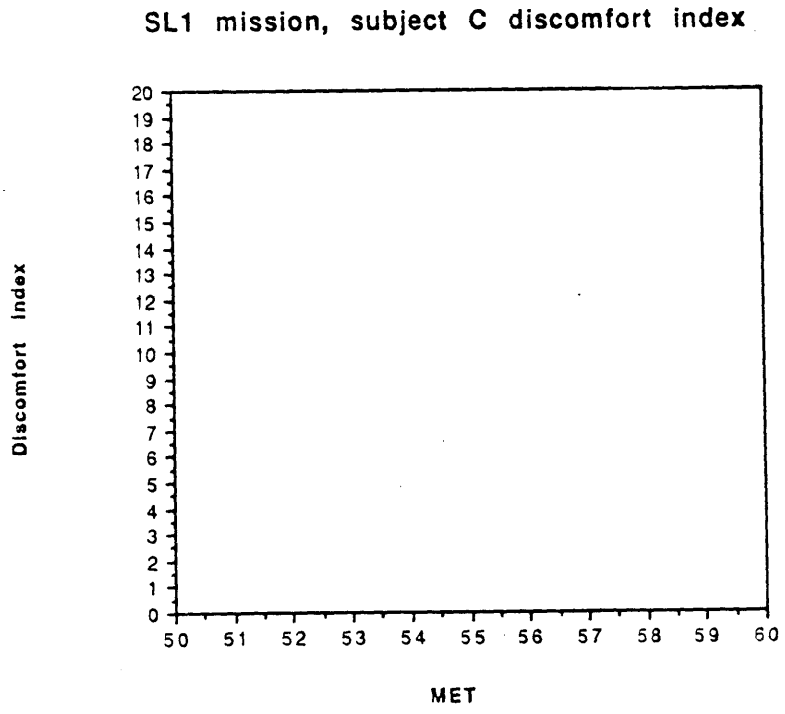

Figure 4.44: Subject C discomfort index

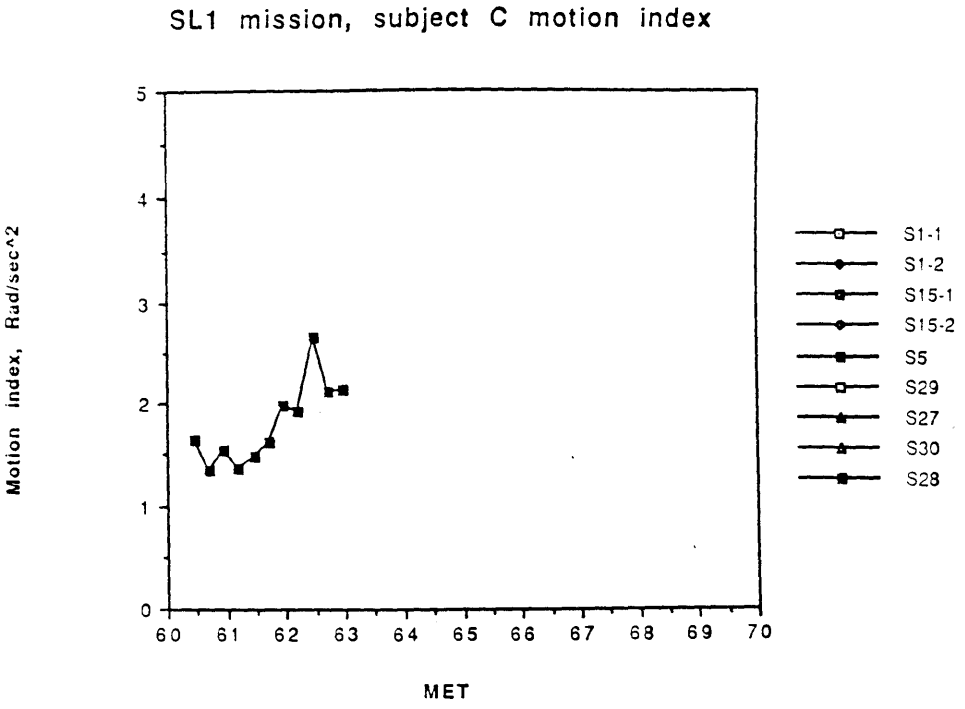

Figure 4.45: Subject C motion index

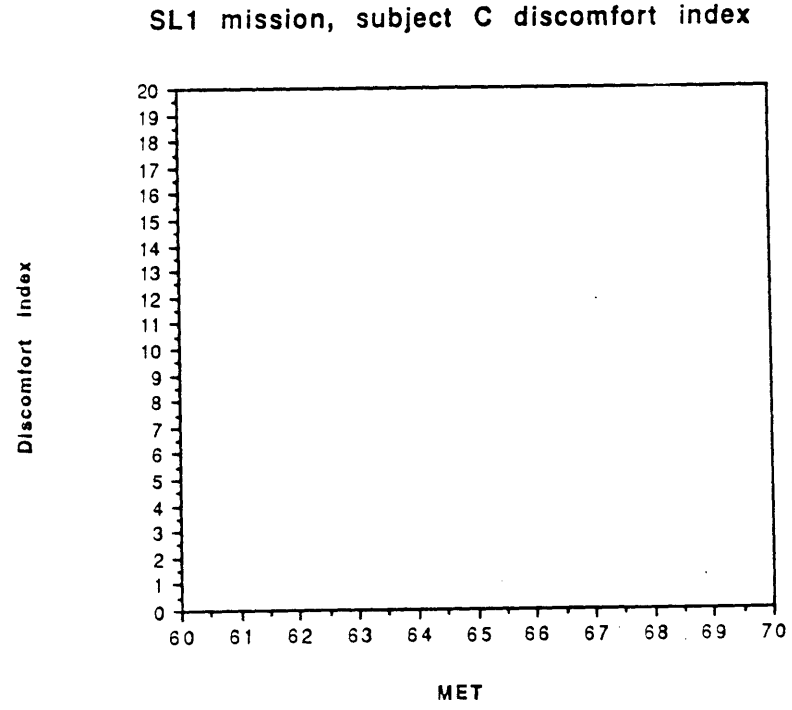

Figure 4.46: Subject C discomfort index

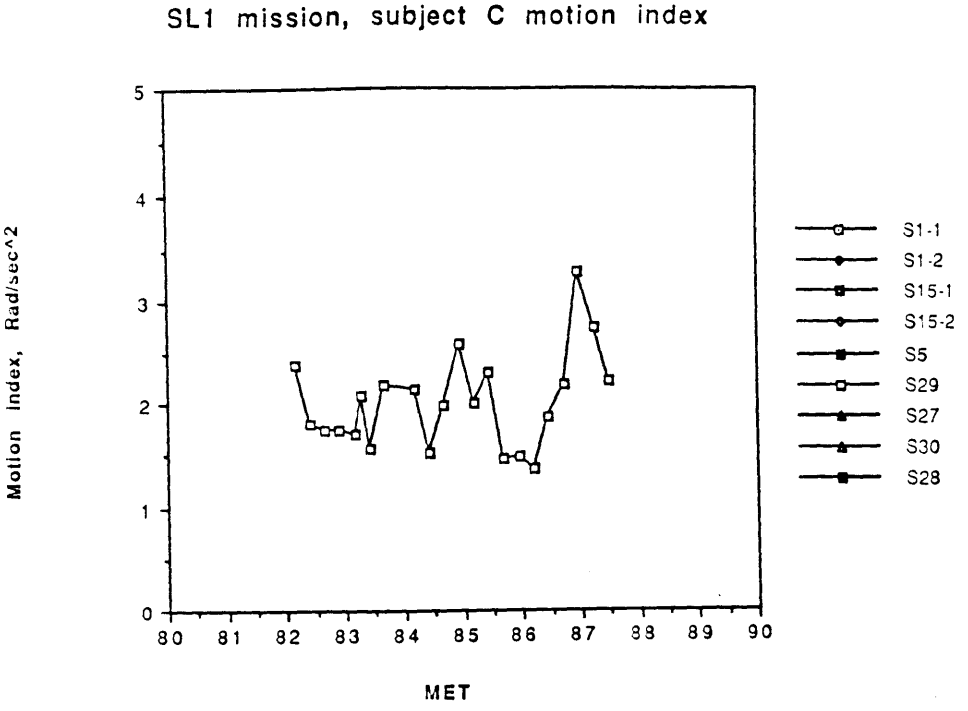

Figure 4.47: Subject C motion index

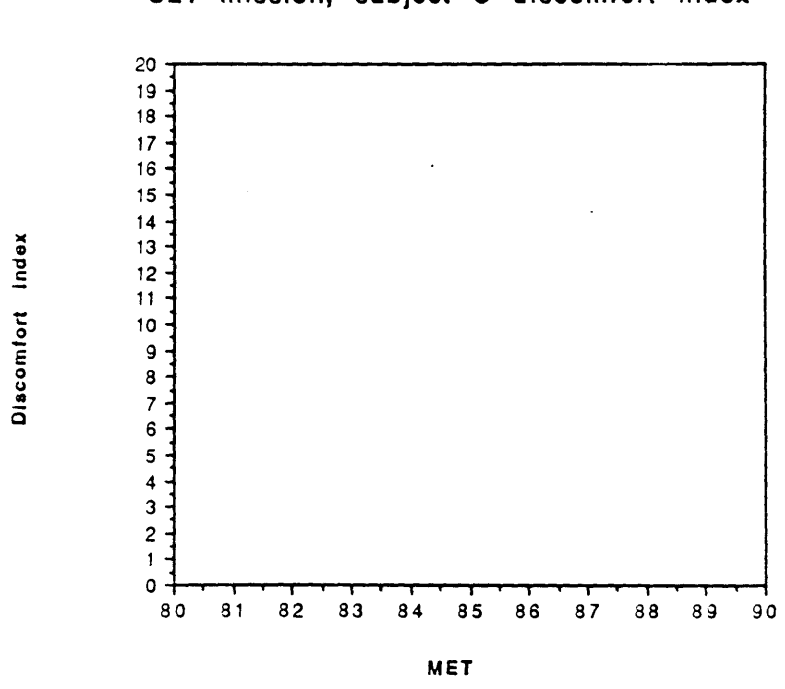

Figure 4.48: Subject C discomfort index

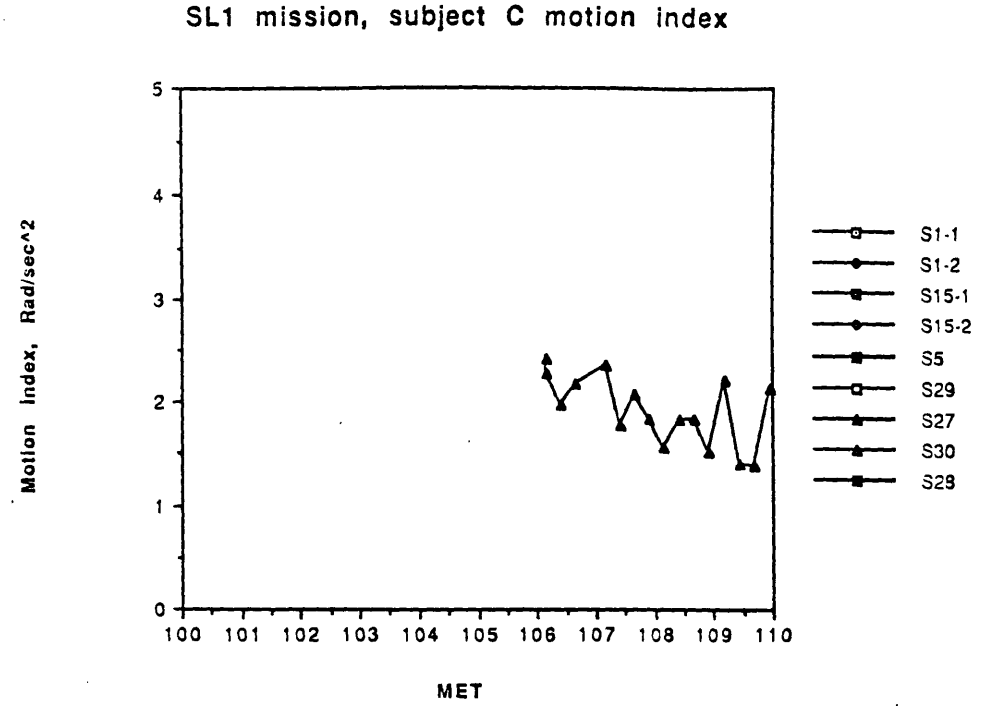

Figure 4.49: Subject C motion index

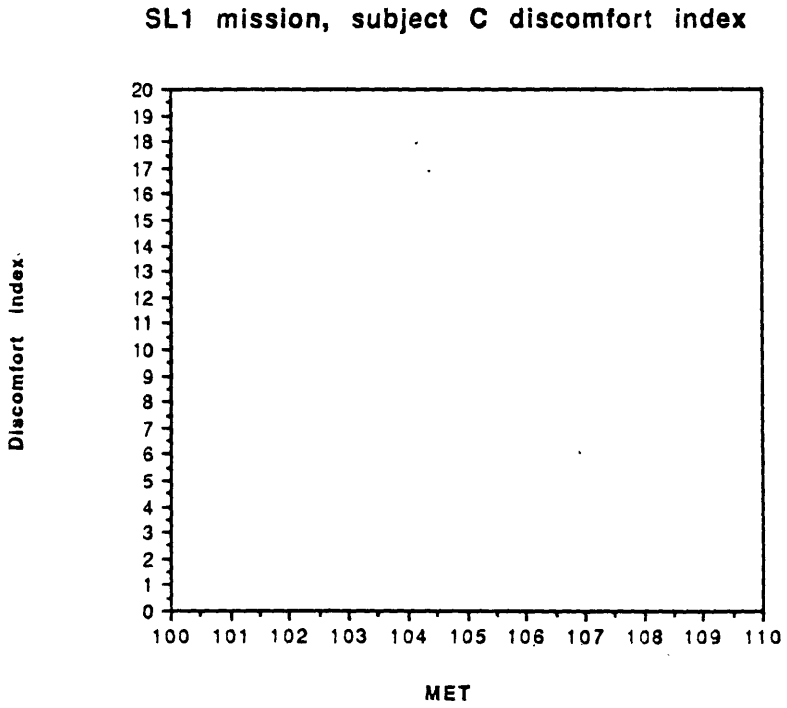

Figure 4.50: Subject C discomfort index

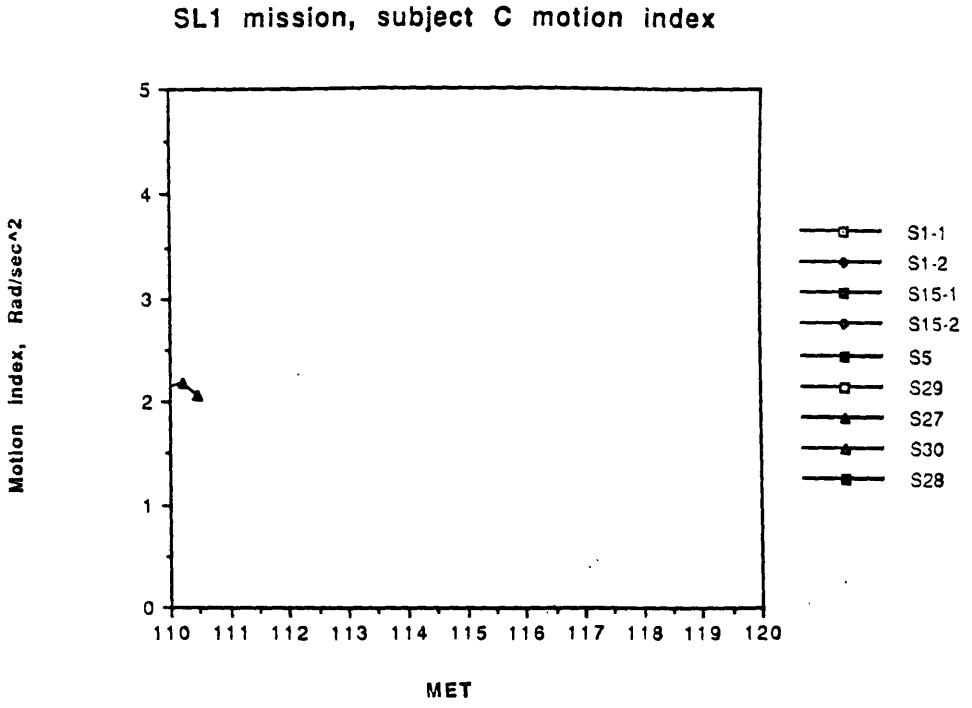

Figure 4.51: Subject C motion index

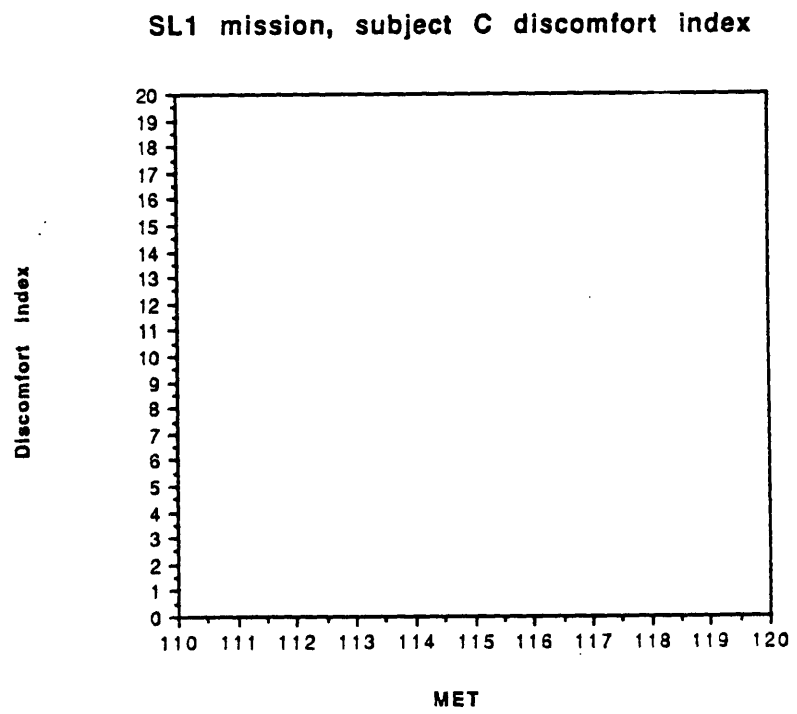

Figure 4.52: Subject C discomfort index

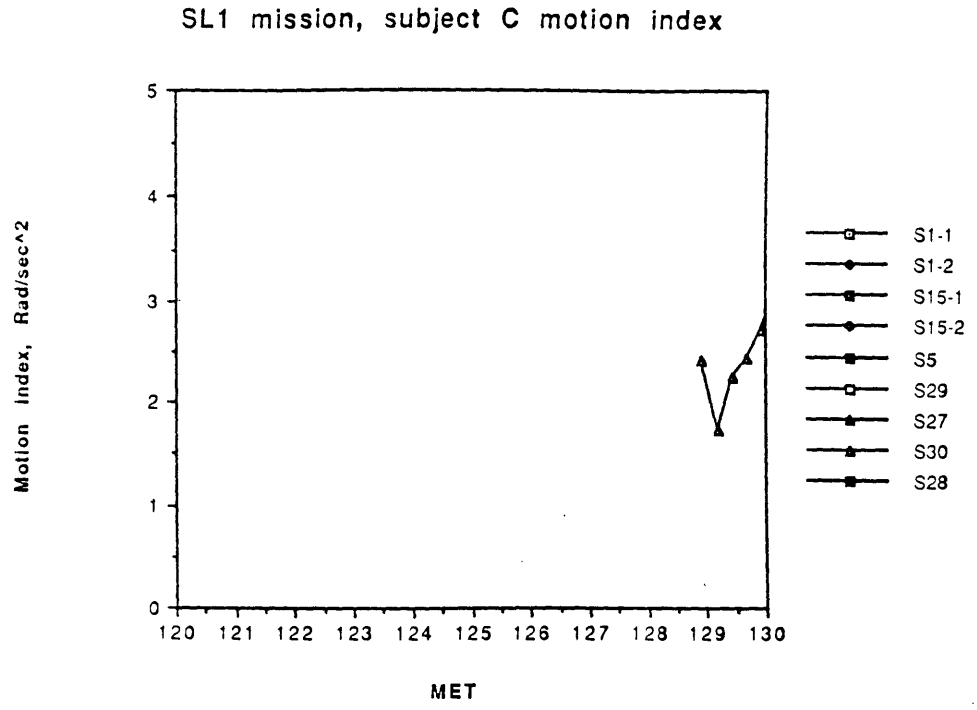

Figure 4.53: Subject C motion index

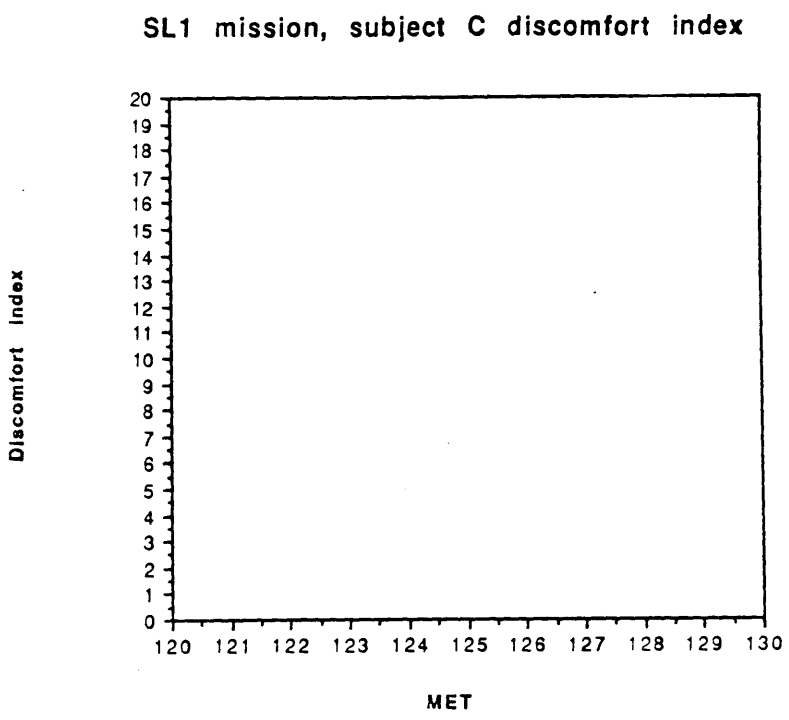

Figure 4.54: Subject C discomfort index

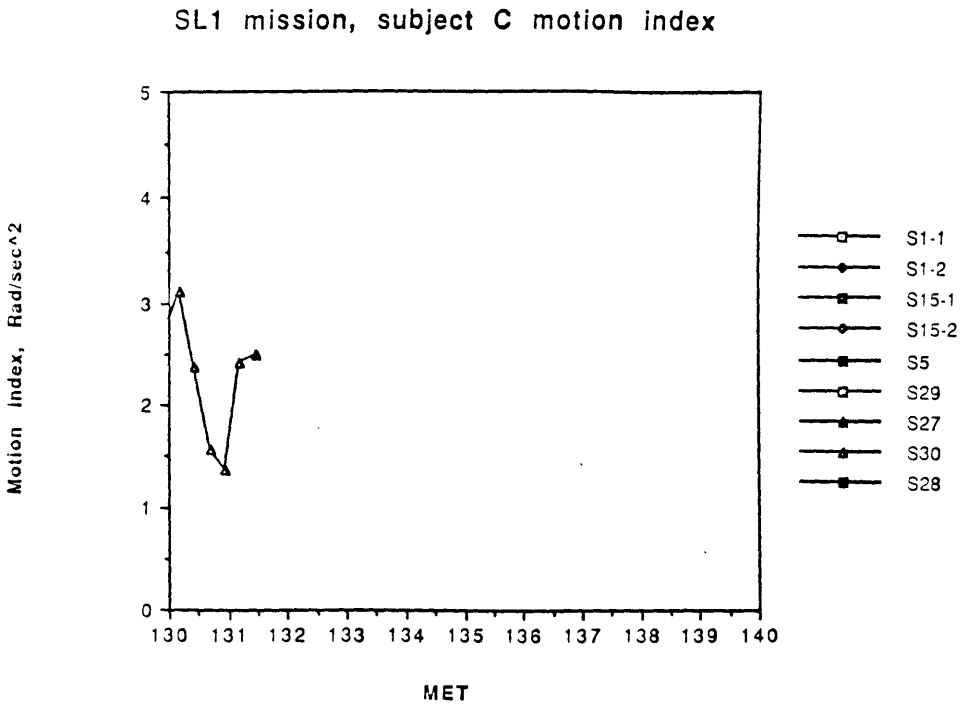

Figure 4.55: Subject C motion index

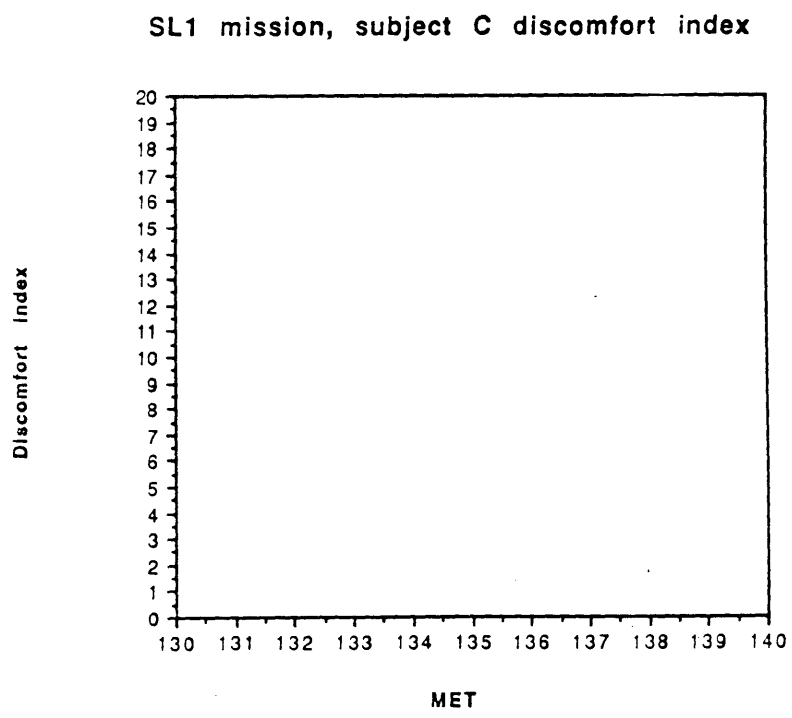

Figure 4.56: Subject C discomfort index

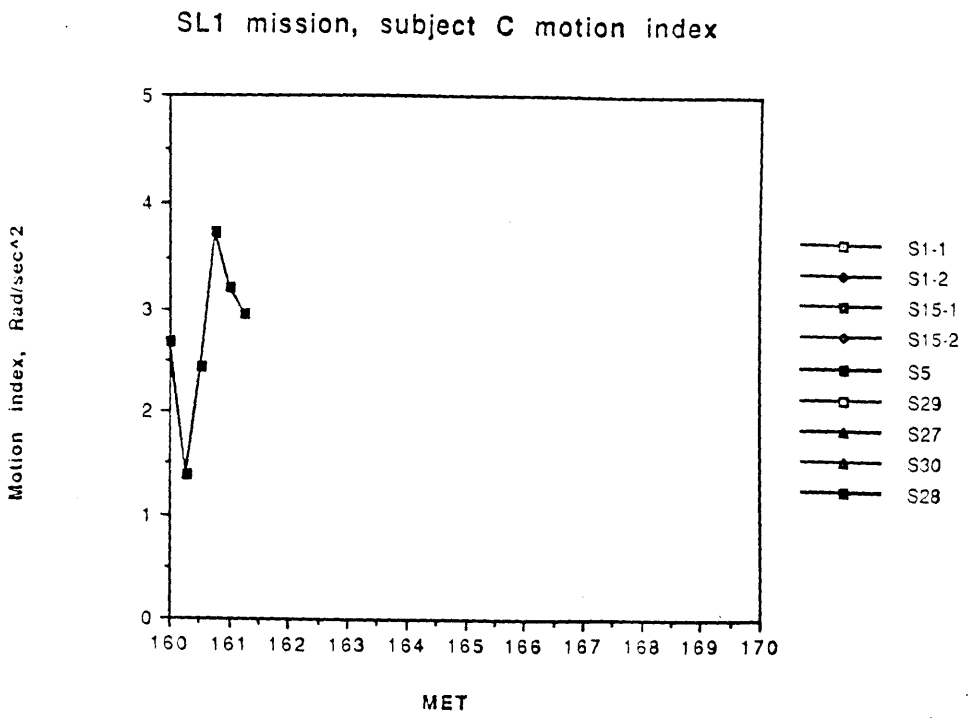

Figure 4.57: Subject C motion index

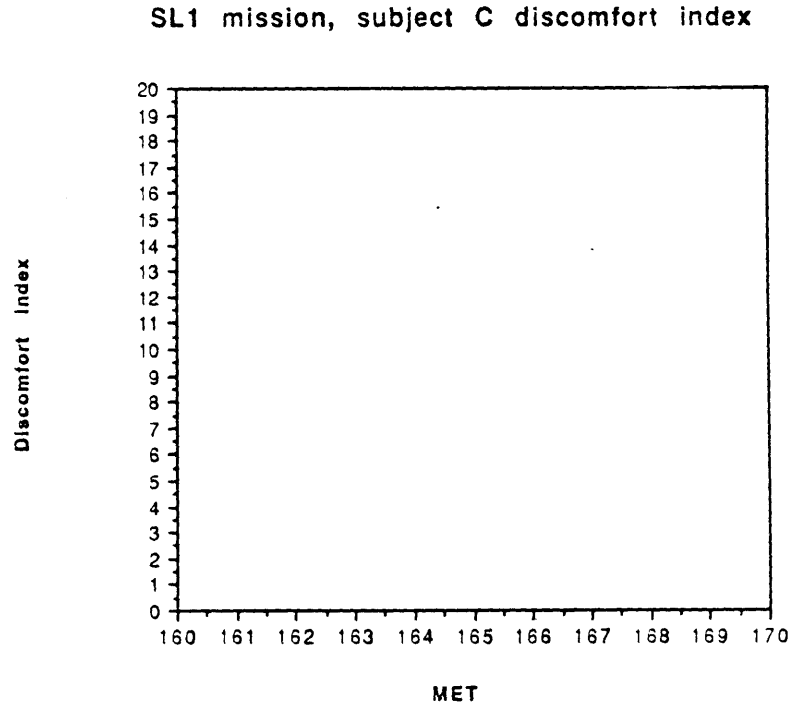

Figure 4.58: Subject C discomfort index

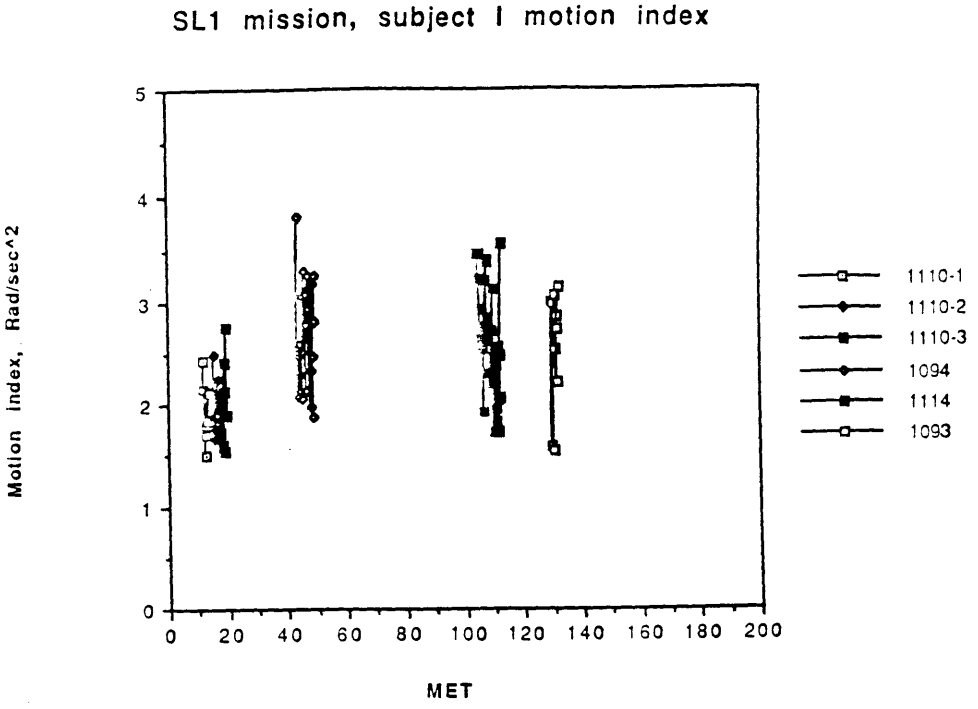

Figure 4.59: Subject I motion index

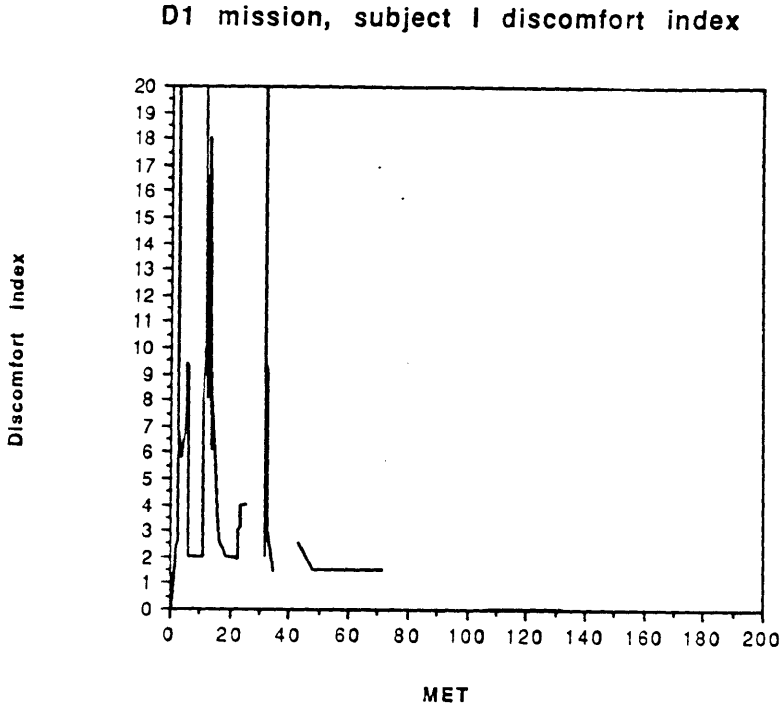

Figure 4.60: Subject I discomfort index

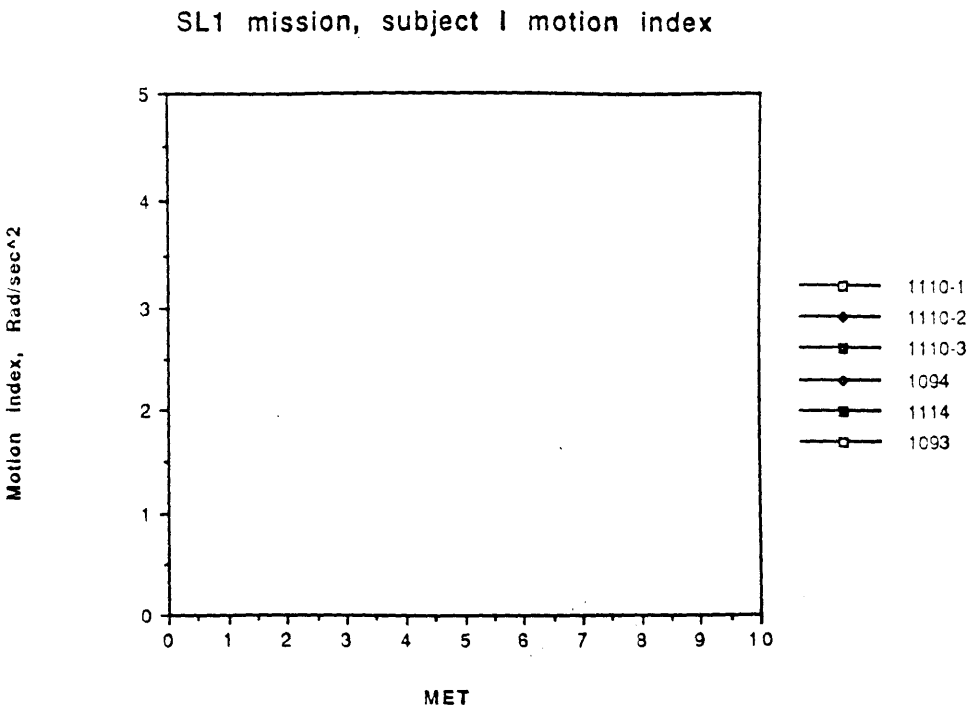

Figure 4.61: Subject I motion index

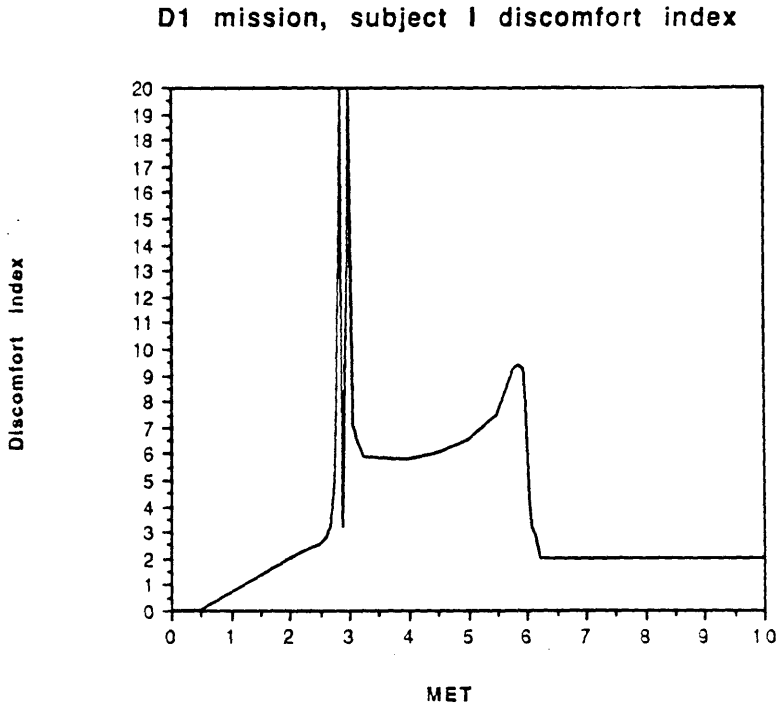

Figure 4.62: Subject I discomfort index

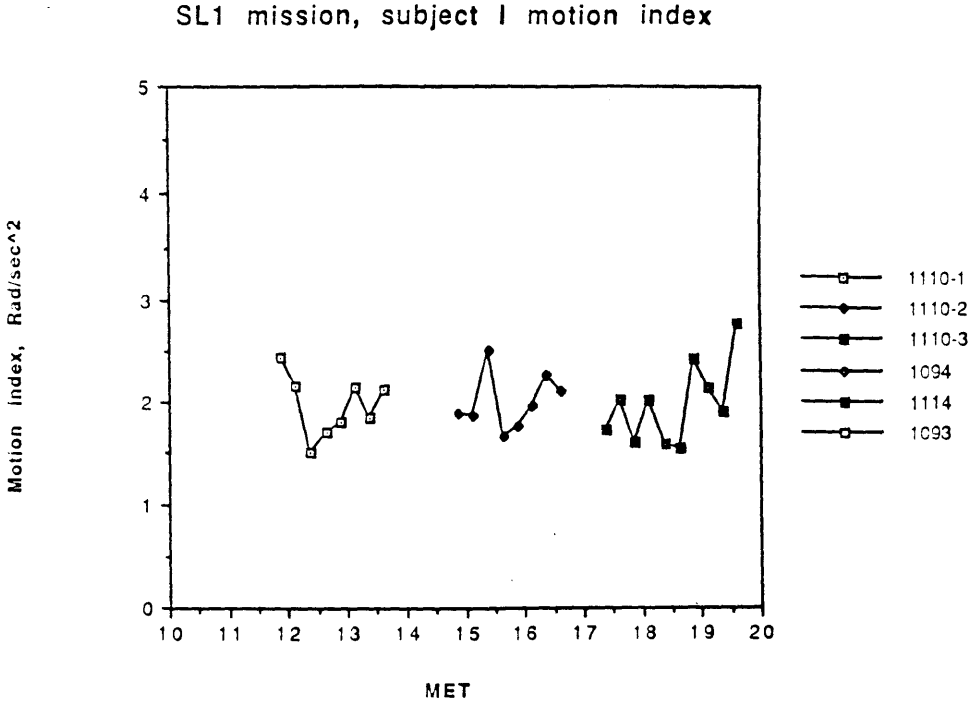

Figure 4.63: Subject I motion index

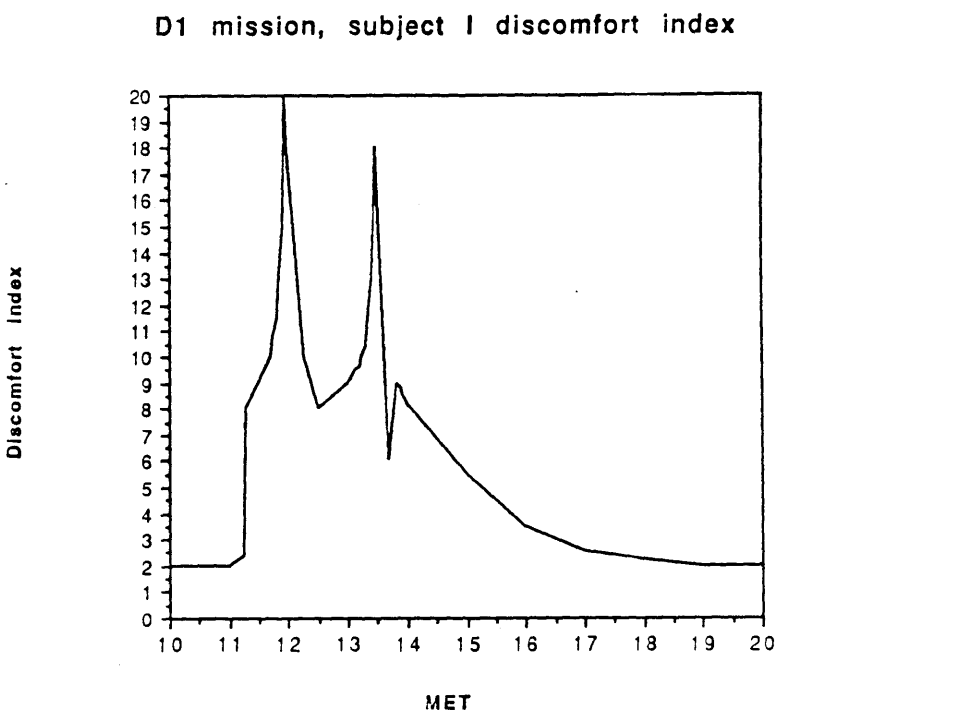

Figure 4.64: Subject I discomfort index

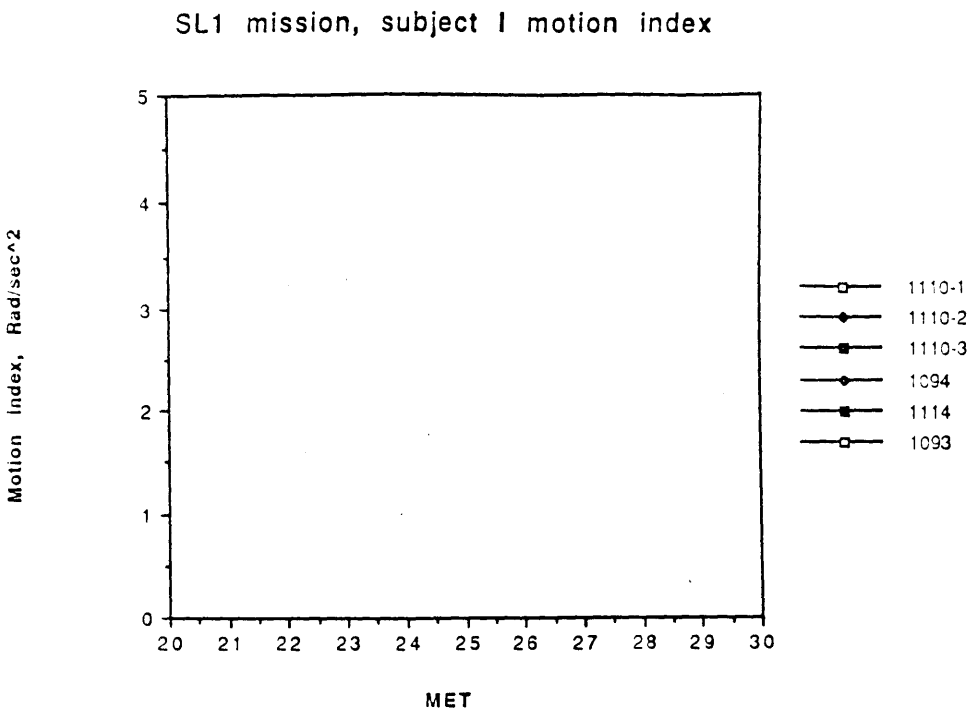

Figure 4.65: Subject I motion index

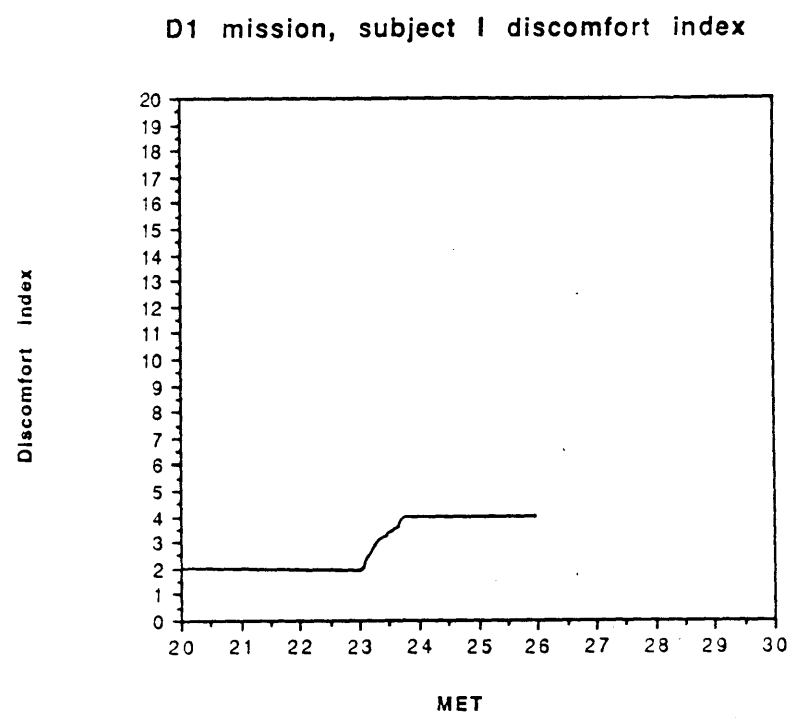

Figure 4.66: Subject I discomfort index

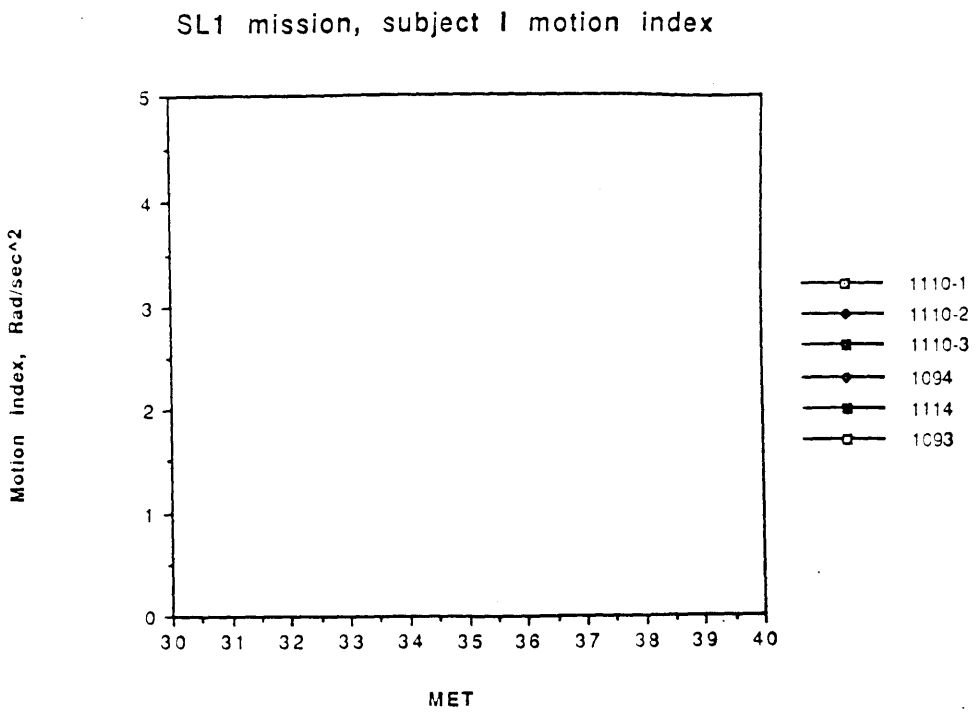

Figure 4.67: Subject I motion index

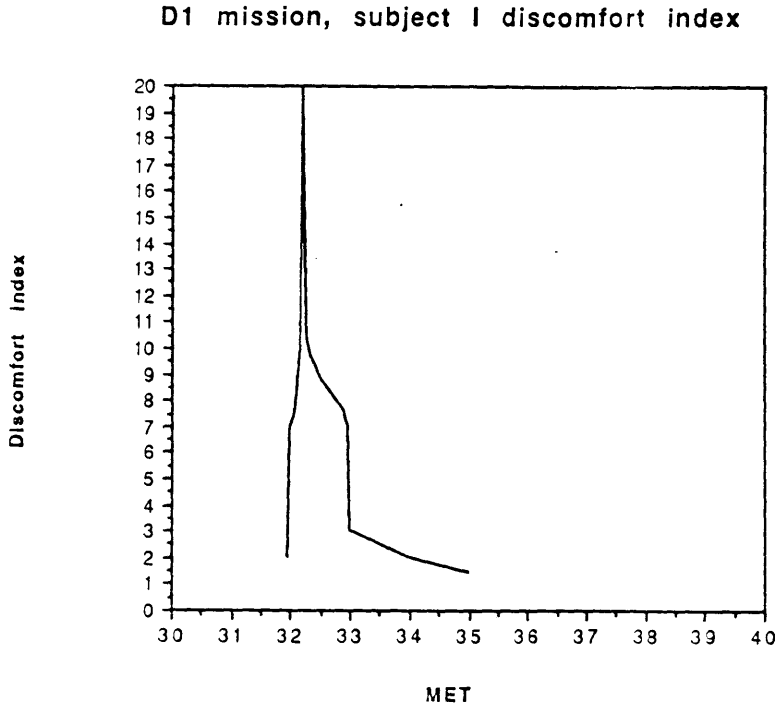

Figure 4.68: Subject I discomfort index

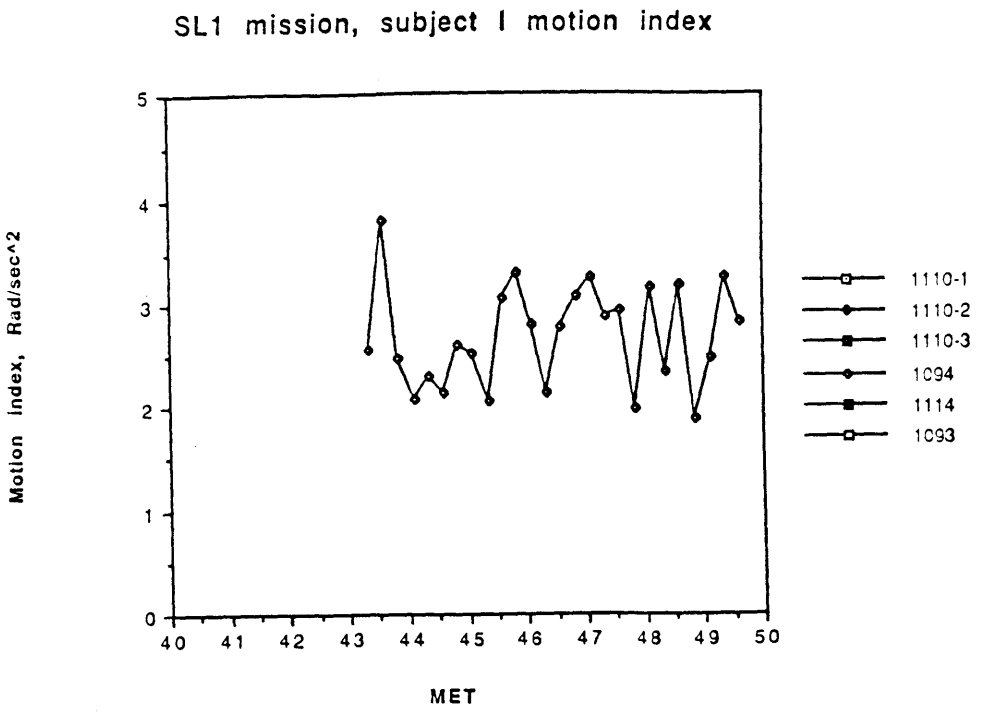

Figure 4.69: Subject I motion index

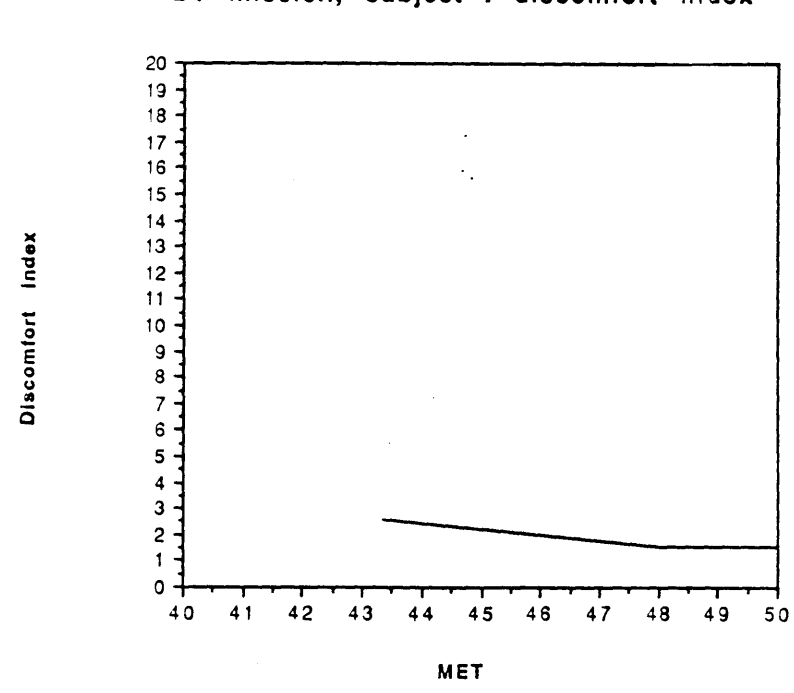

Figure 4.70: Subject I discomfort index

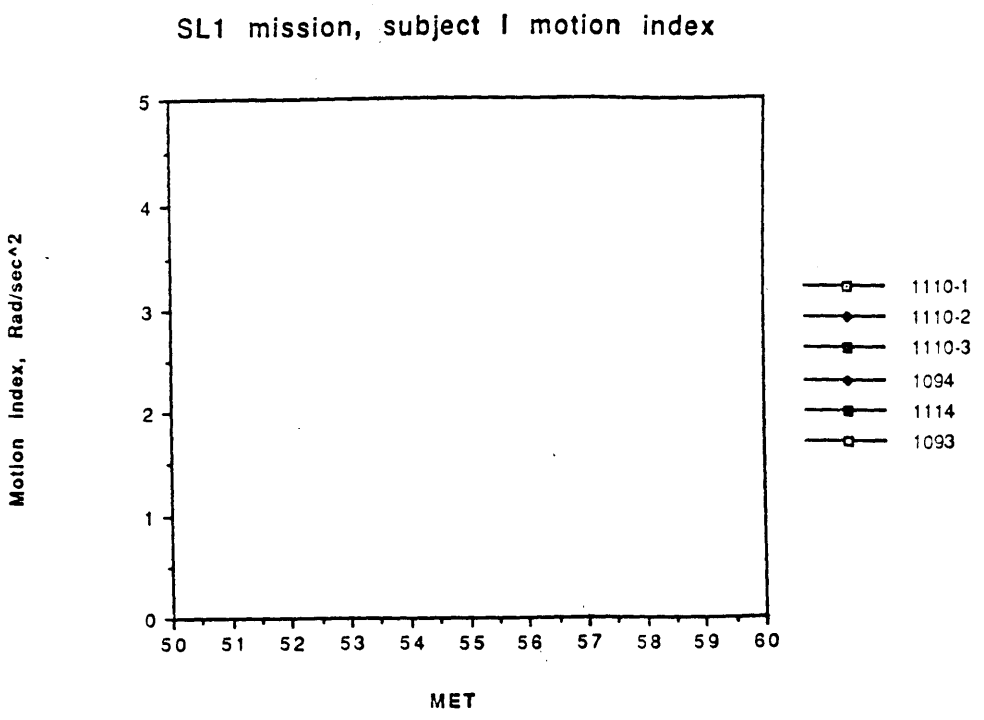

Figure 4.71: Subject I motion index

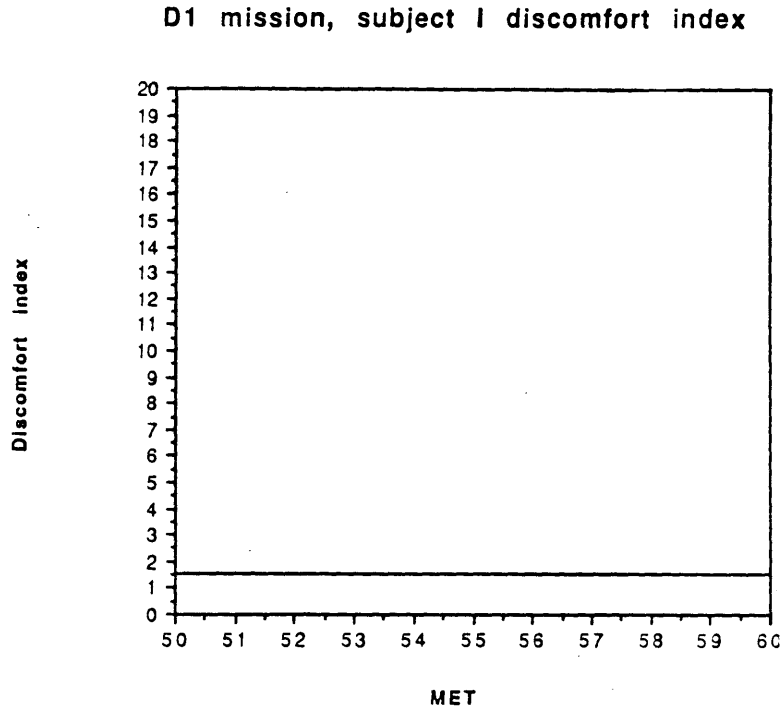

Figure 4.72: Subject I discomfort index

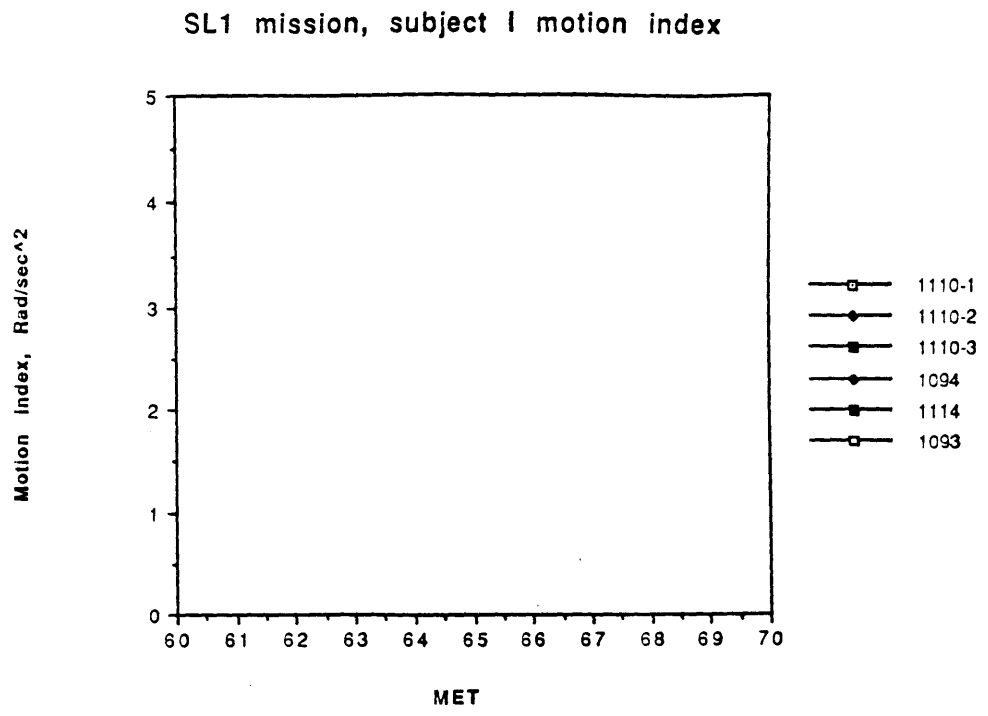

Figure 4.73: Subject I motion index

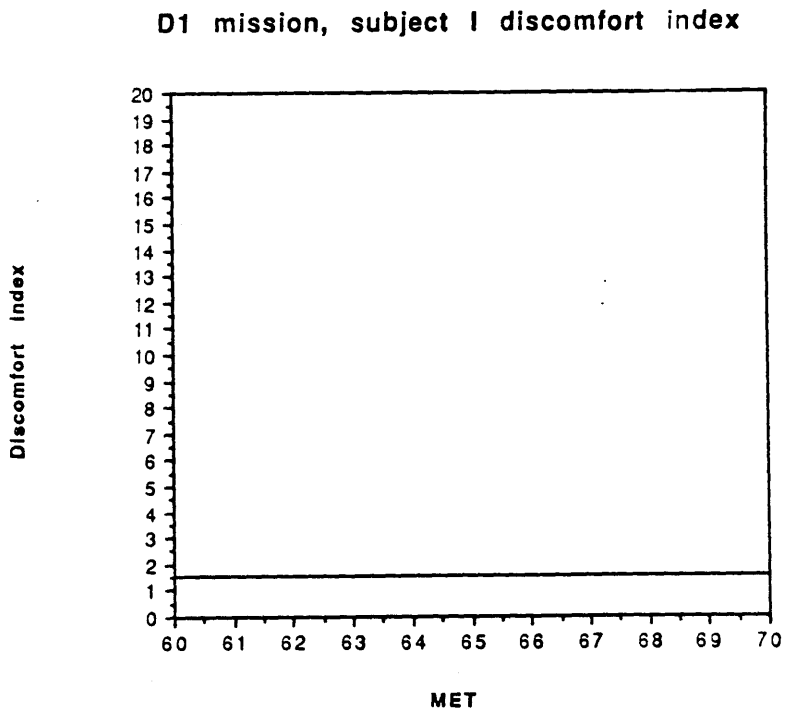

Figure 4.74: Subject I discomfort index

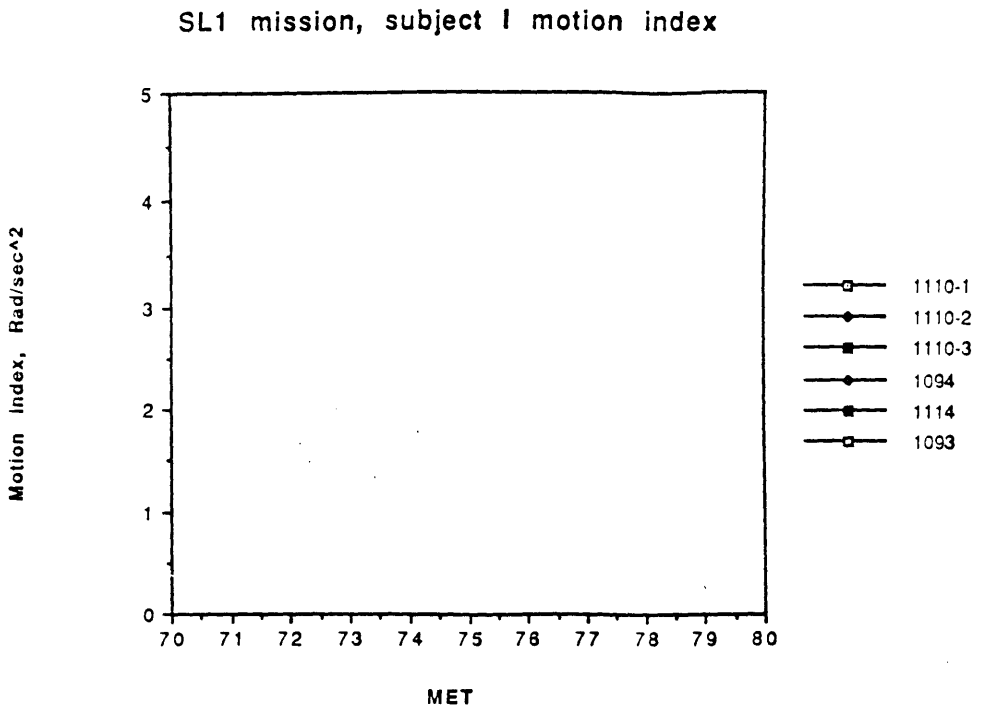

Figure 4.75: Subject I motion index

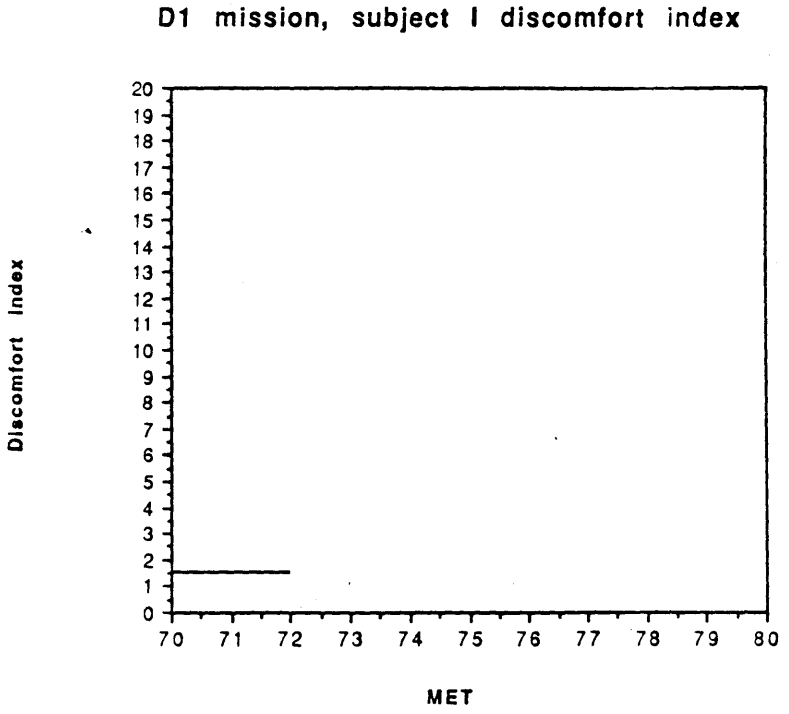

Figure 4.76: Subject I discomfort index

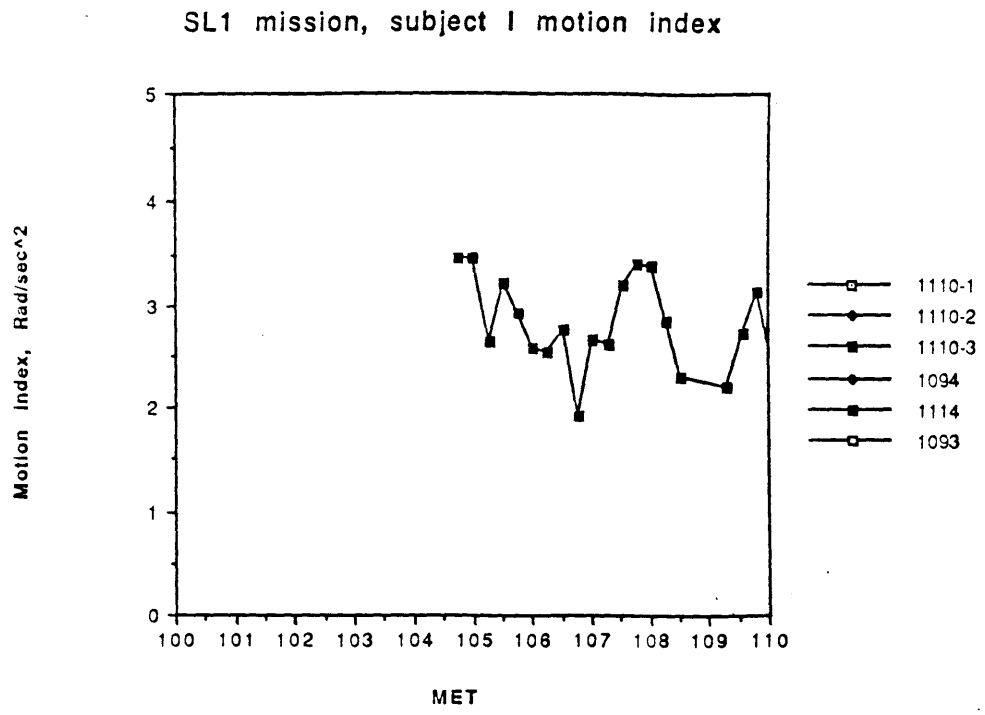

Figure 4.77: Subject I motion index

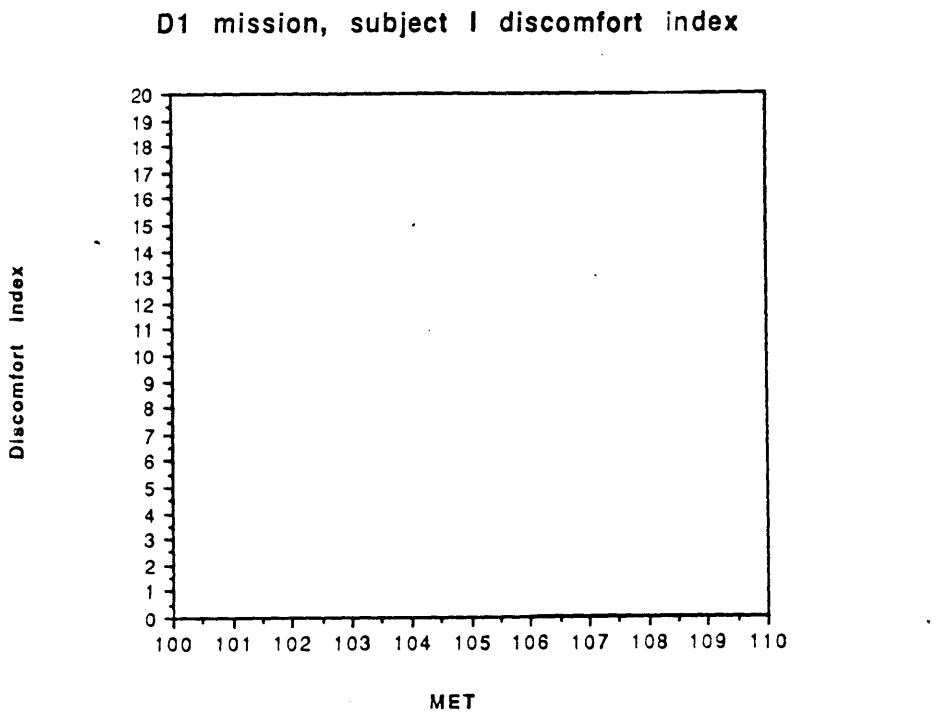

Figure 4.78: Subject I discomfort index

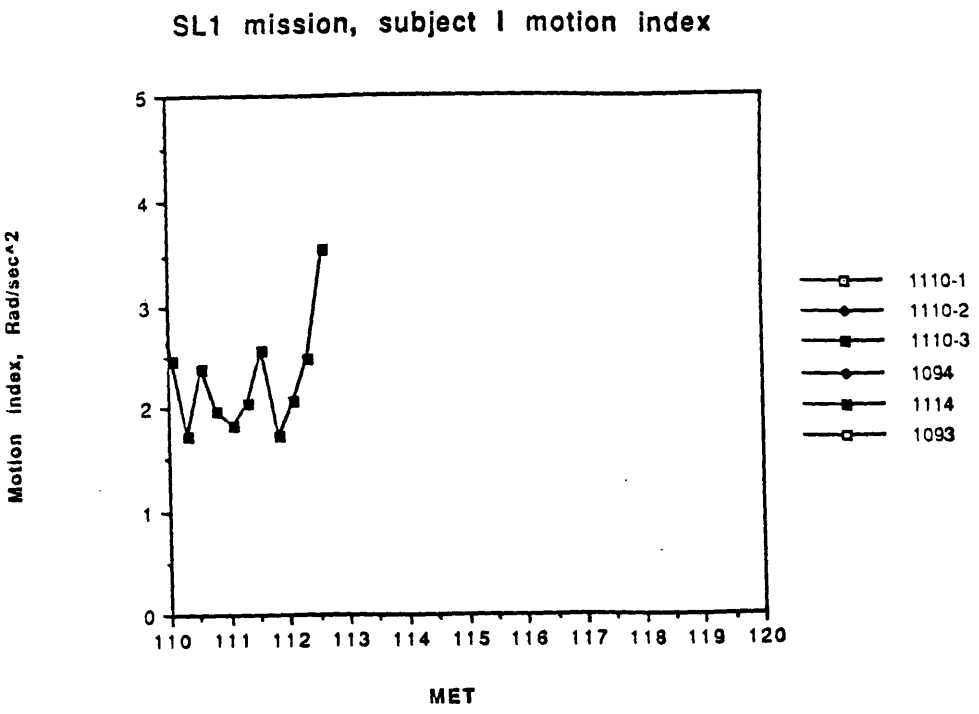

Figure 4.79: Subject I motion index

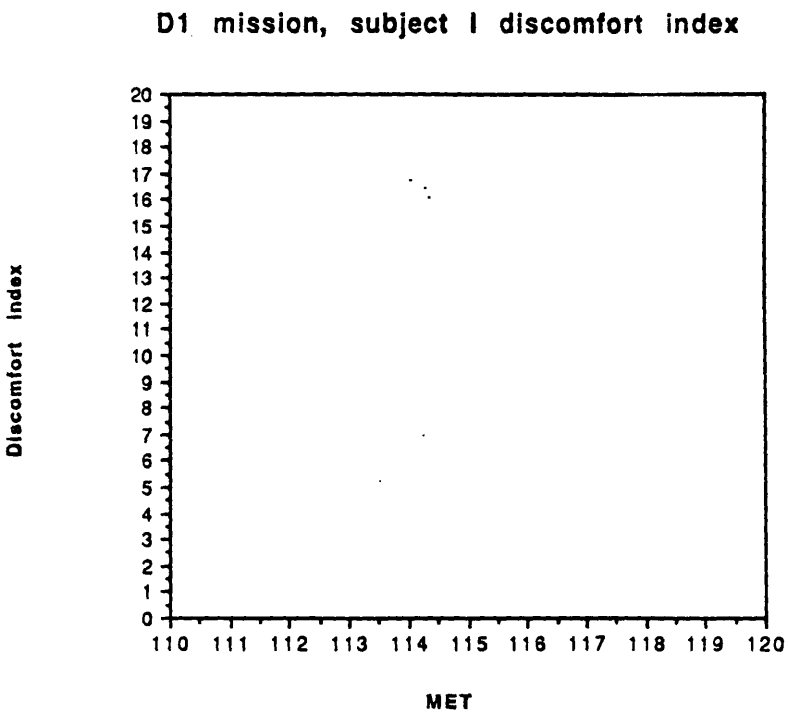

Figure 4.80: Subject I discomfort index

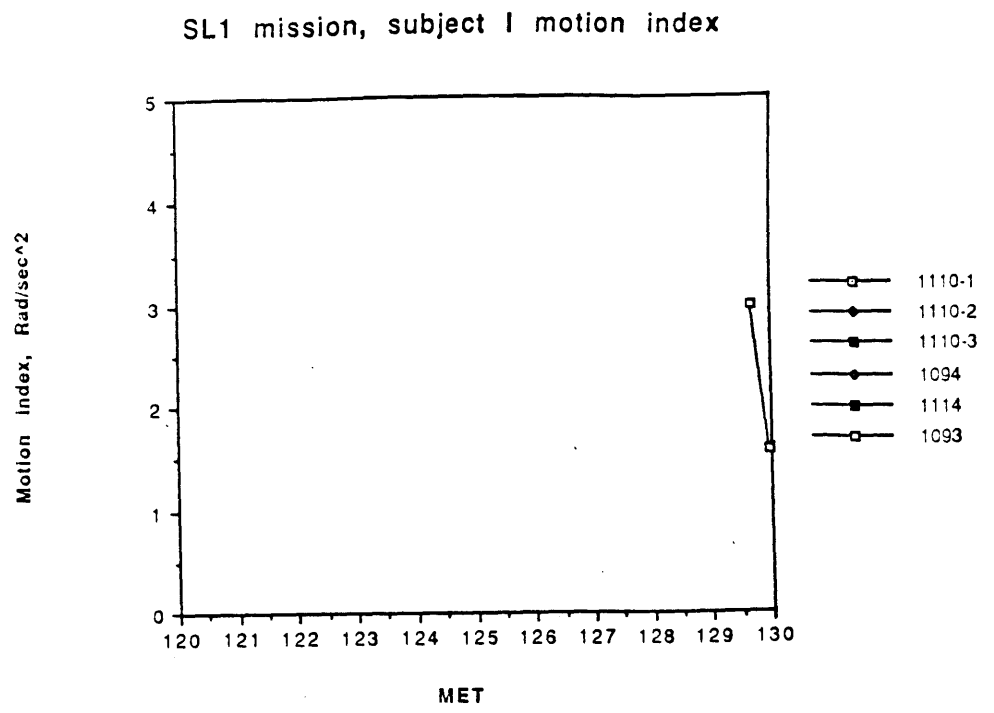

Figure 4.81: Subject I motion index

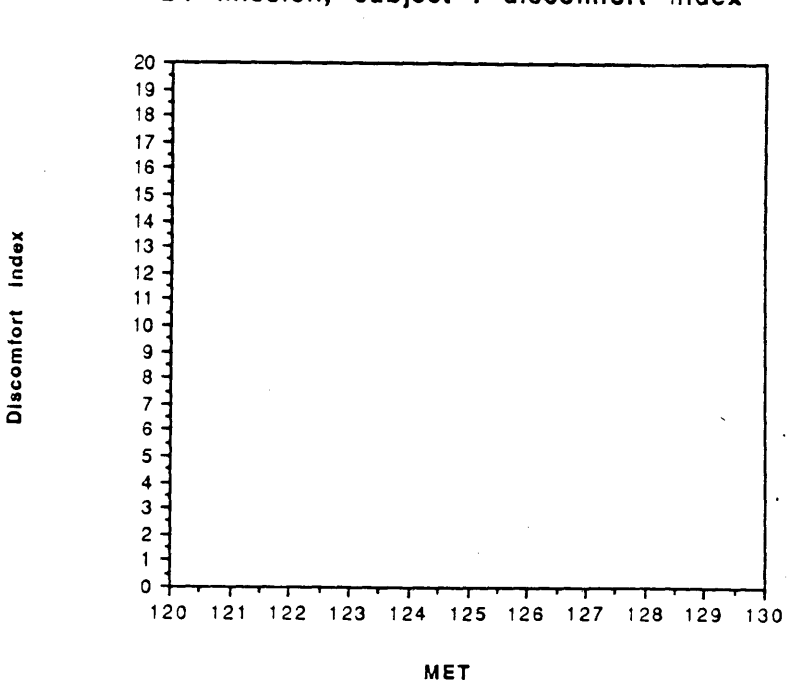

Figure 4.82: Subject I discomfort index

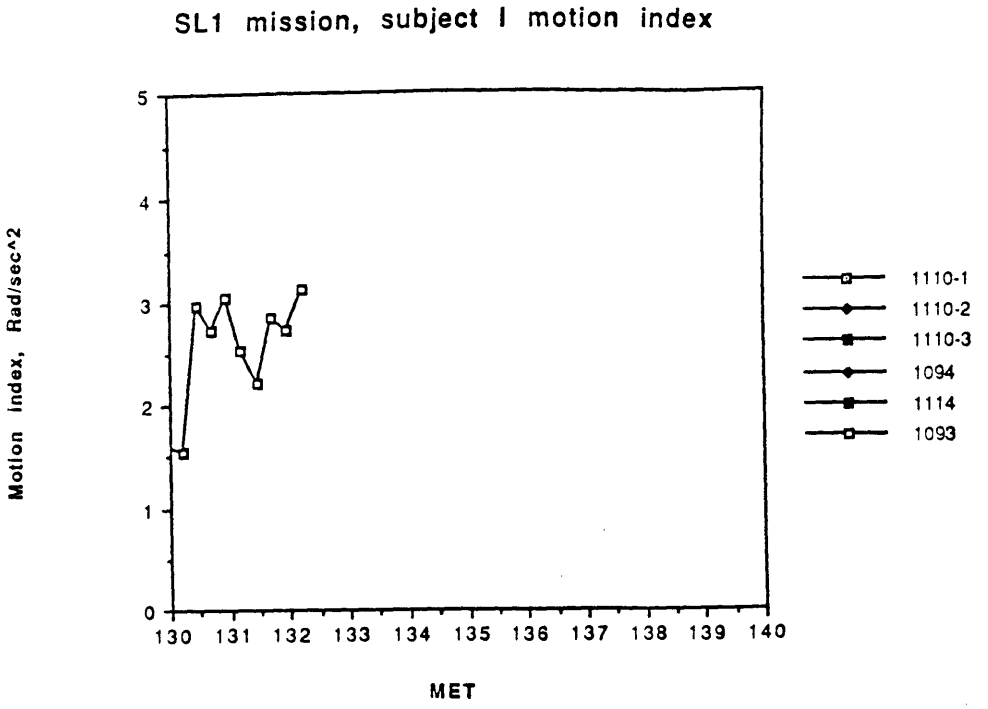

Figure 4.83: Subject I motion index

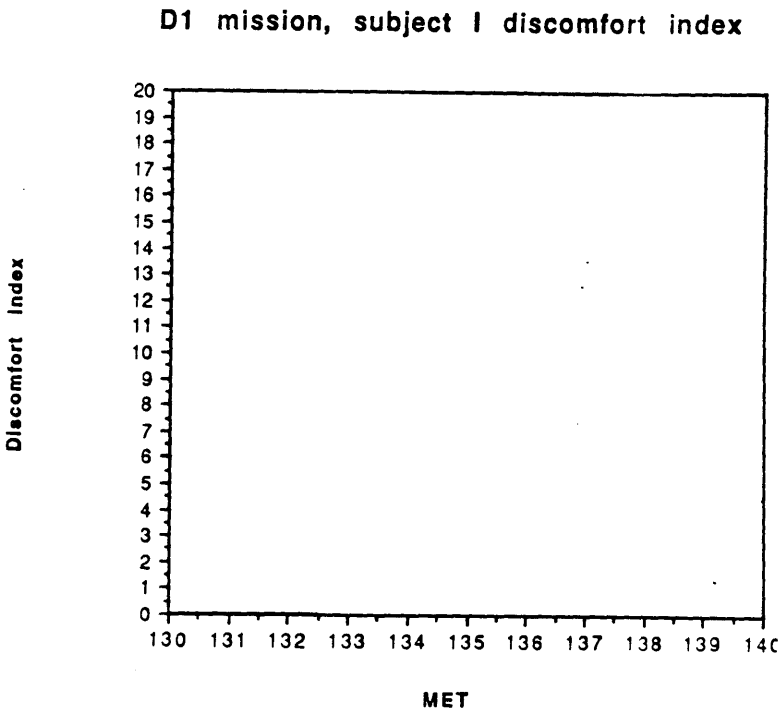

Figure 4.84: Subject I discomfort index

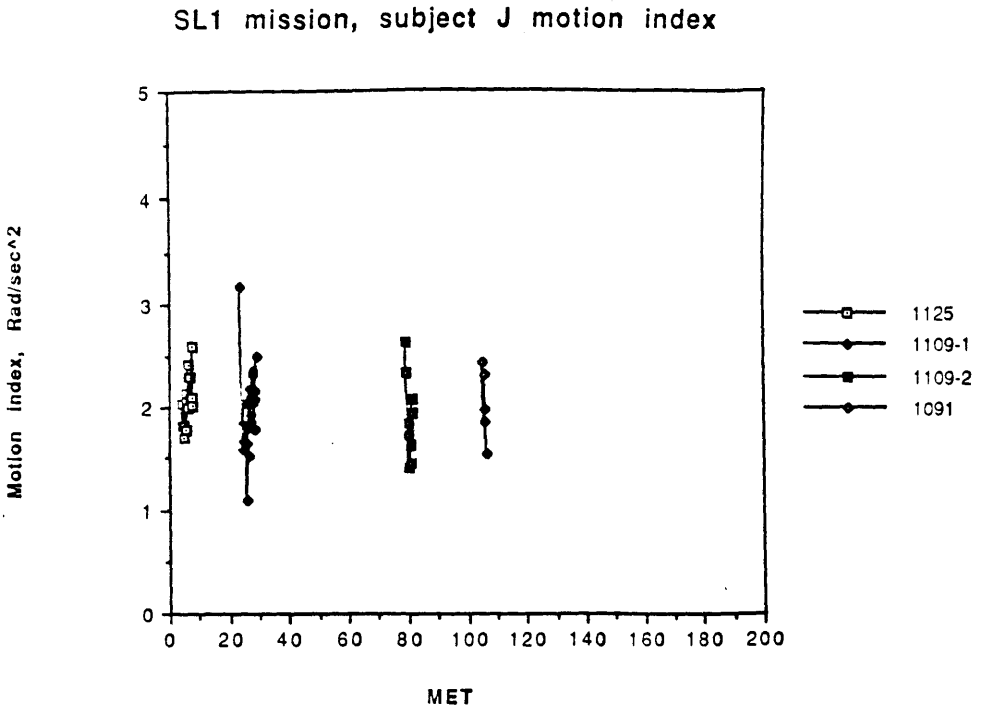

Figure 4.85: Subject J motion index

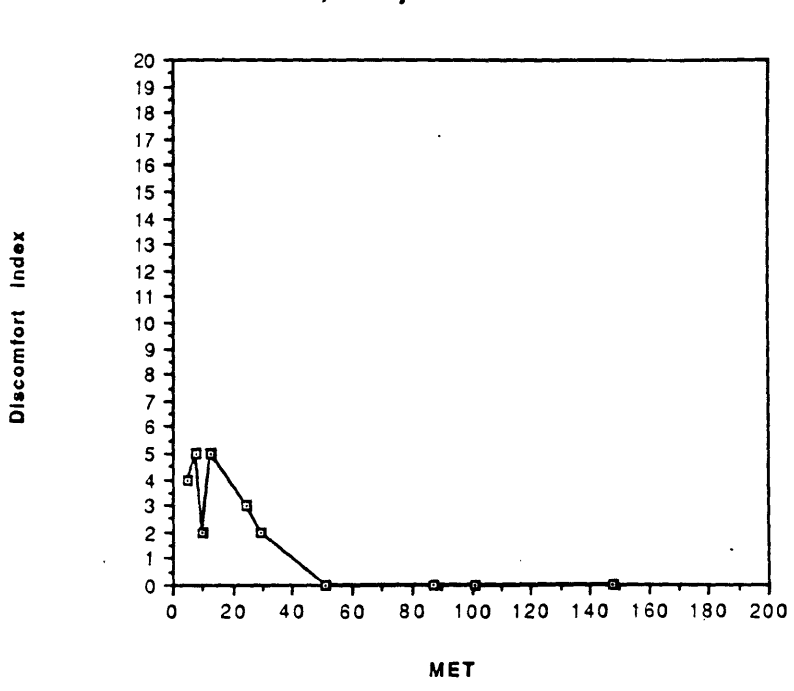

Figure 4.86: Subject J discomfort index

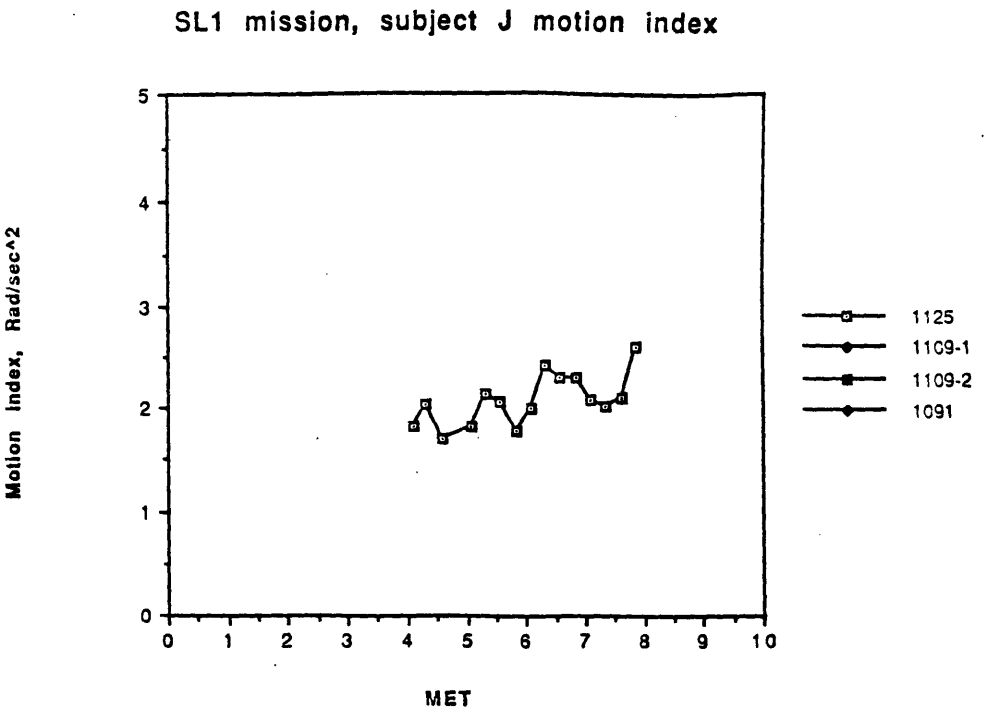

Figure 4.87: Subject J motion index

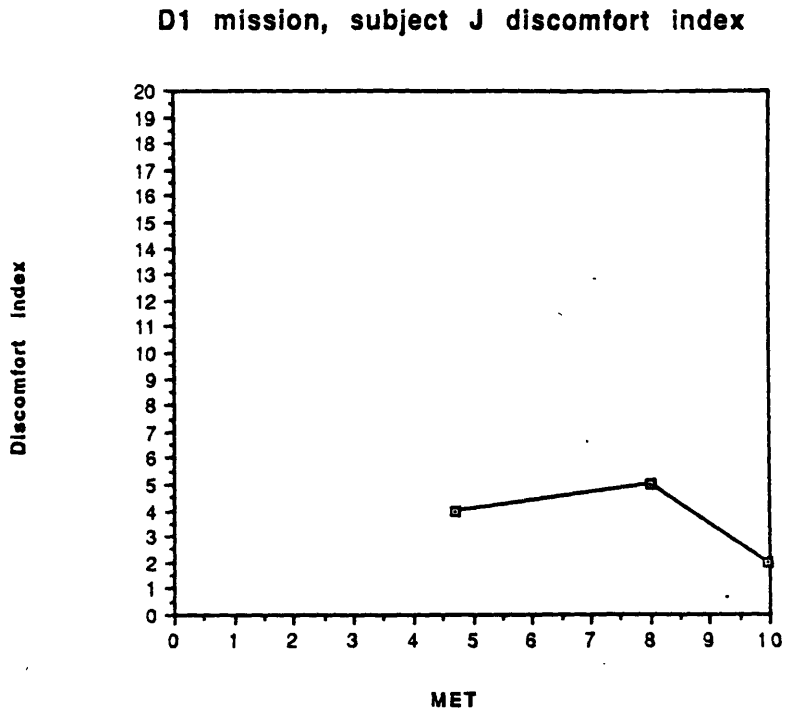

Figure 4.88: Subject J discomfort index

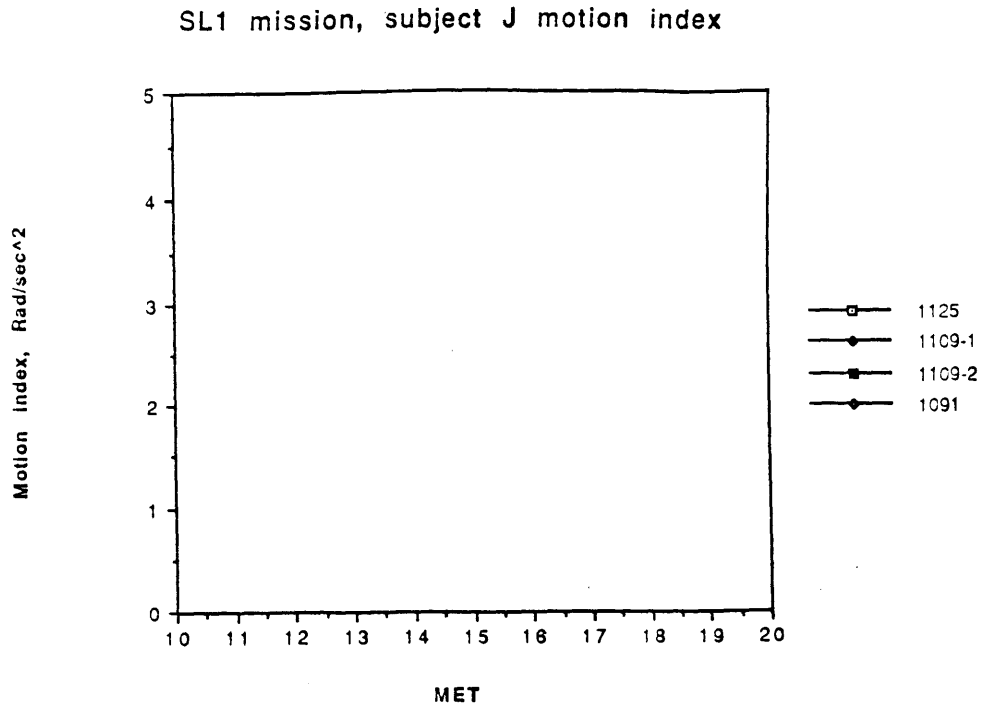

Figure 4.89: Subject J motion index

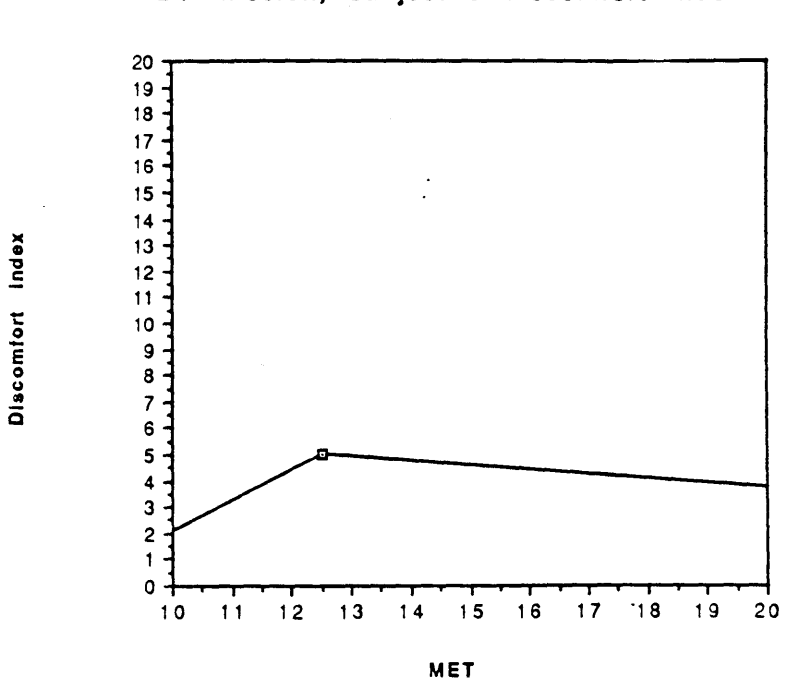

Figure 4.90: Subject J discomfort index

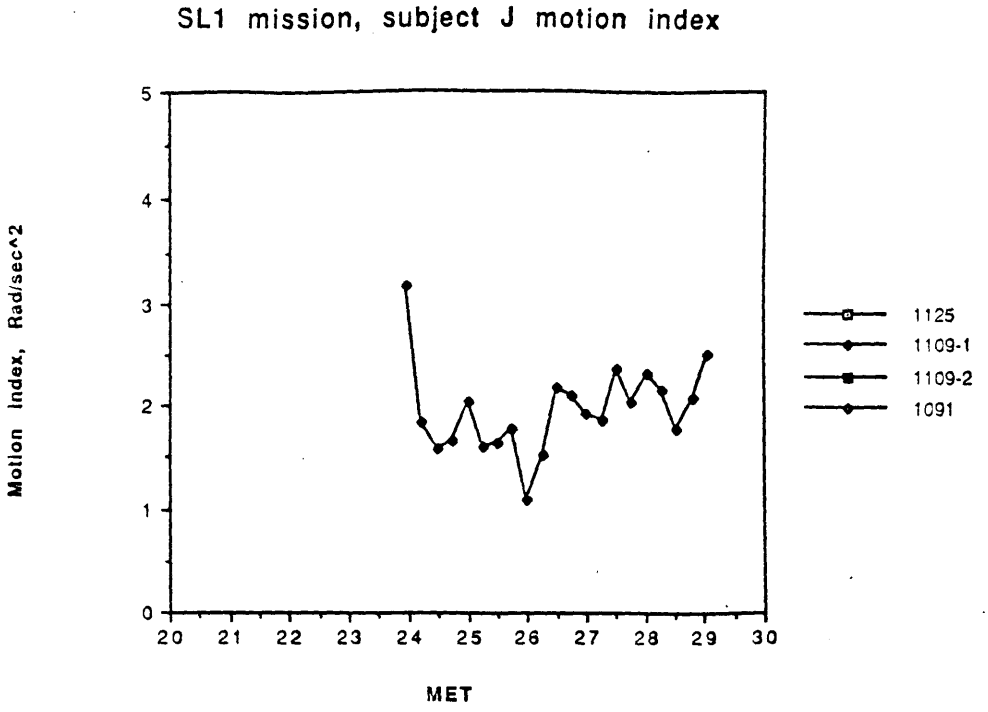

Figure 4.91: Subject J motion index

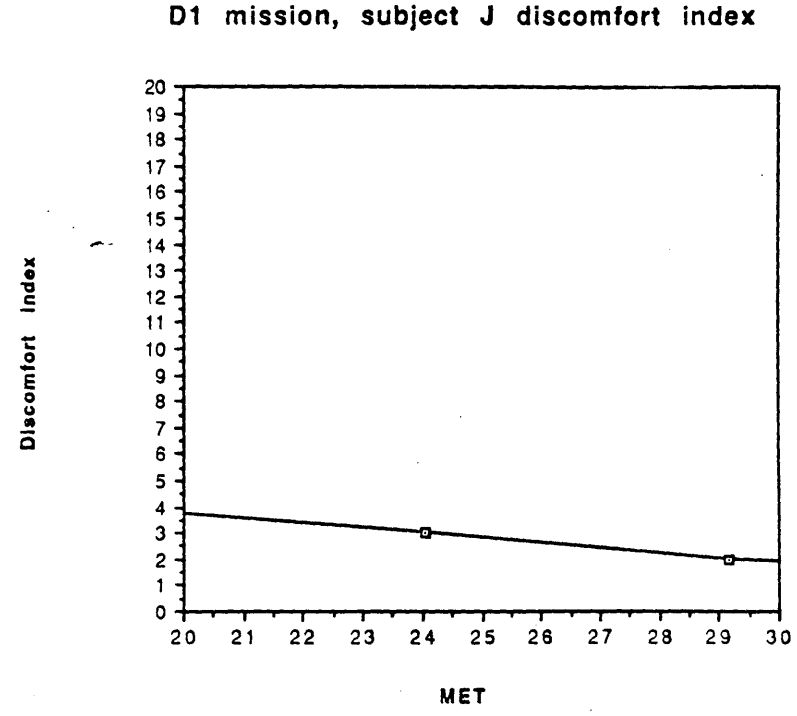

Figure 4.92: Subject J discomfort index

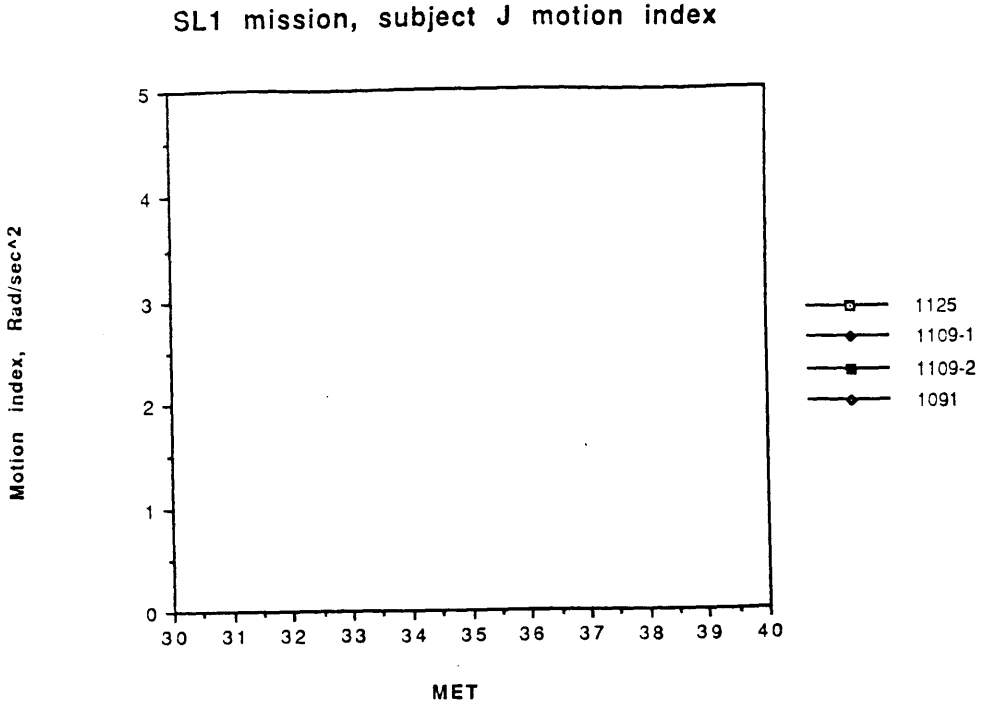

Figure 4.93: Subject J motion index

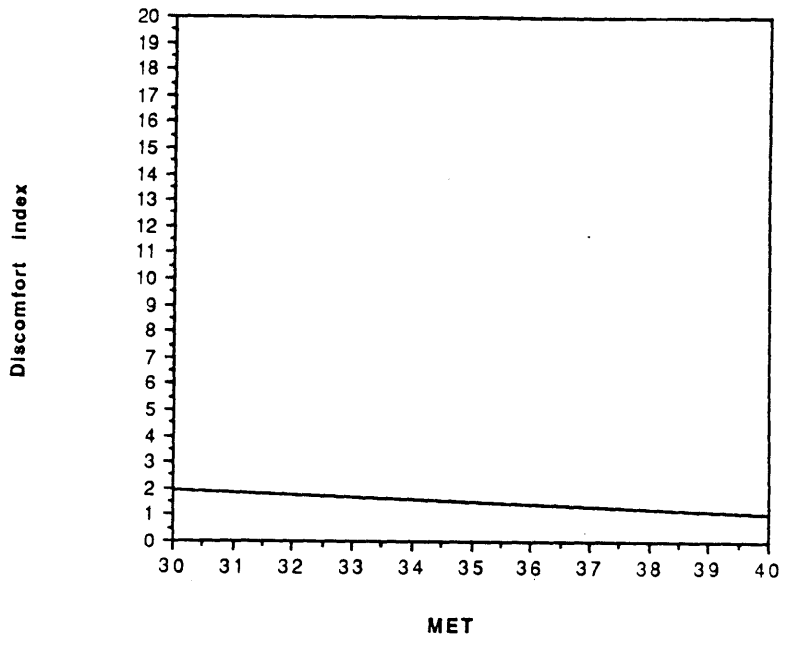

Figure 4.94: Subject J discomfort index

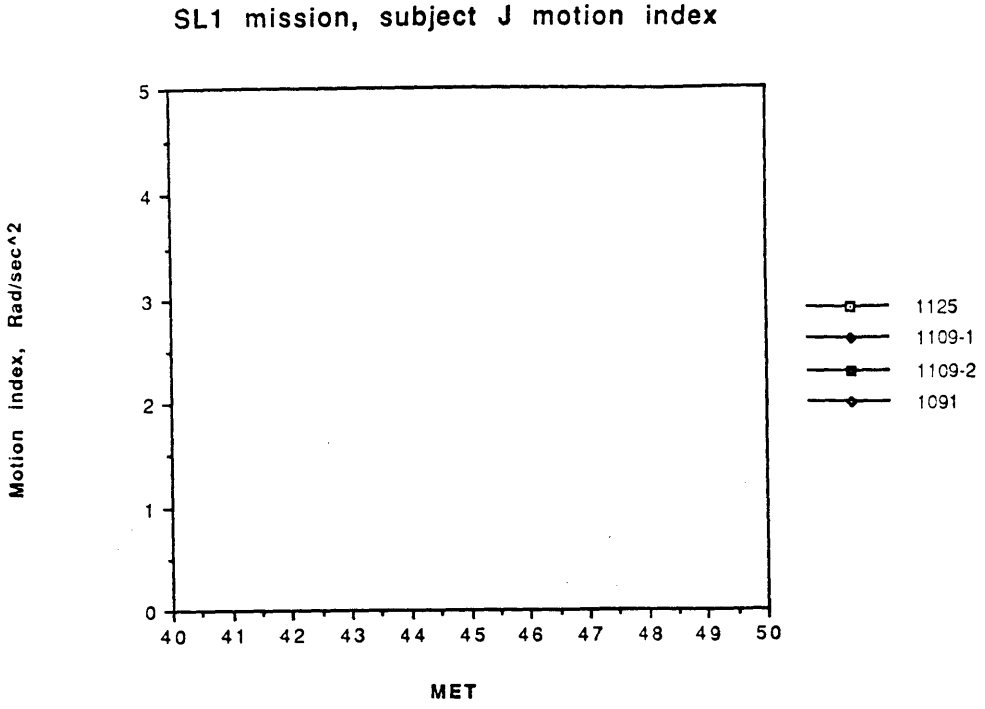

Figure 4.95: Subject J motion index

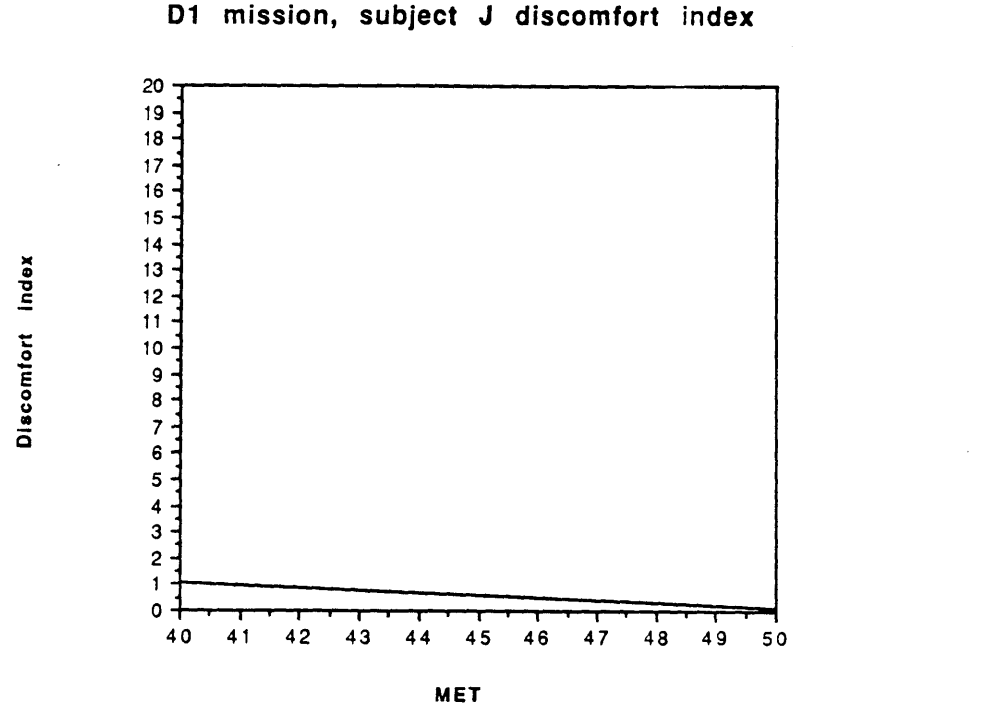

Figure 4.96: Subject J discomfort index

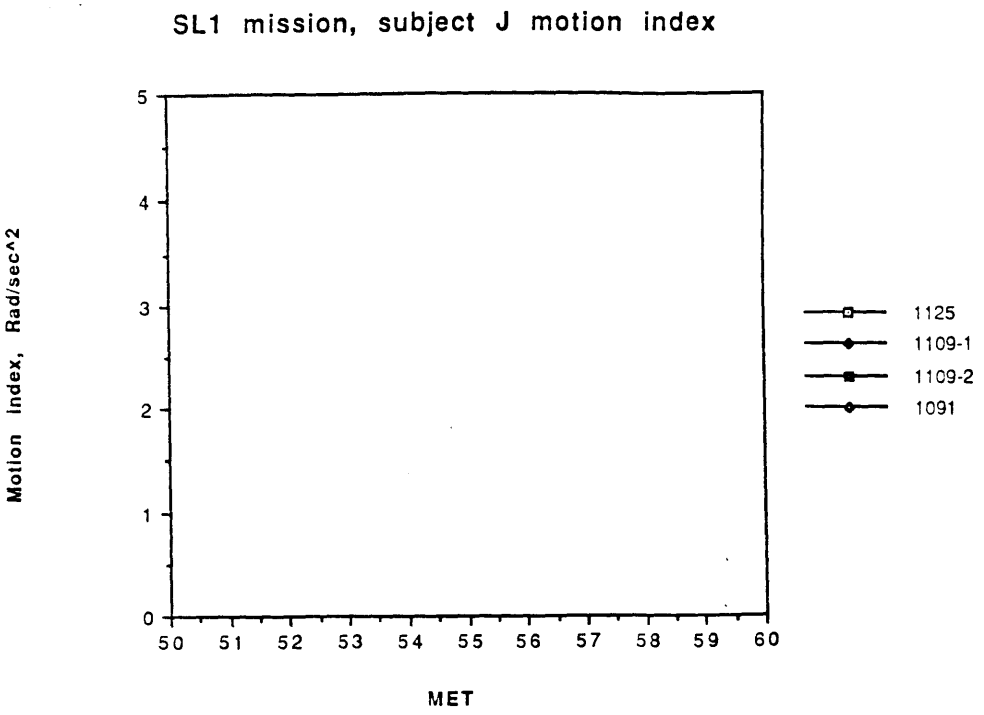

Figure 4.97: Subject J motion index

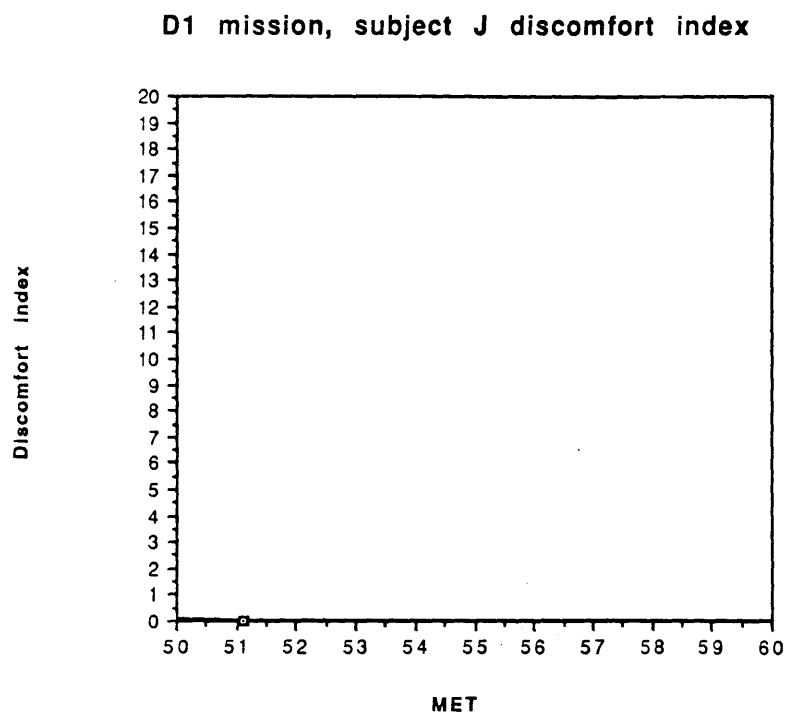

Figure 4.98: Subject J discomfort index

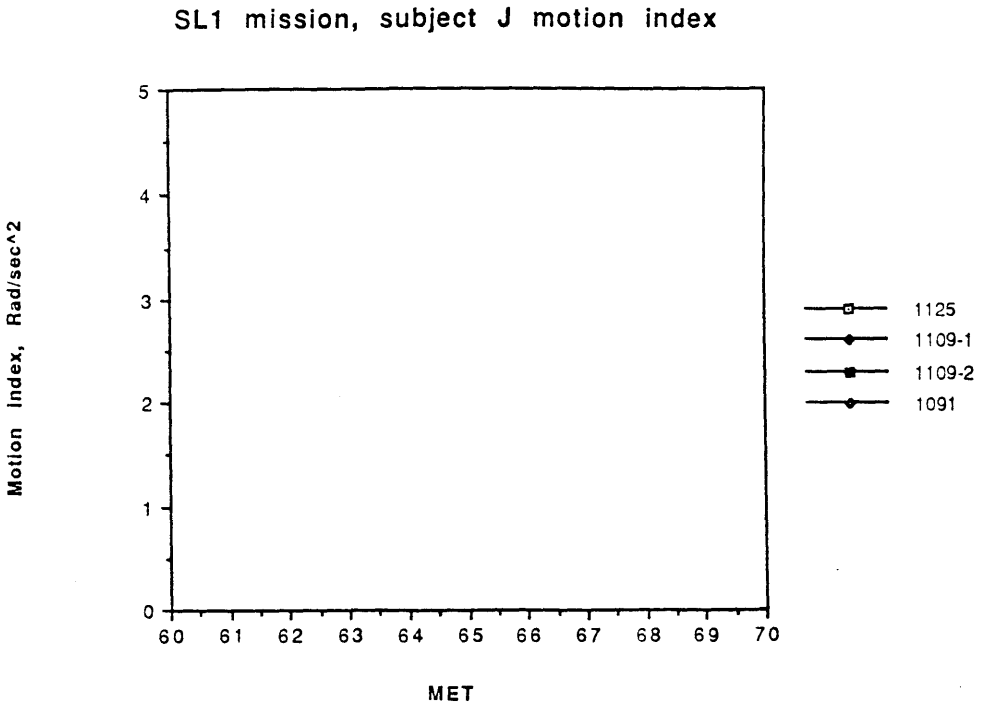

Figure 4.99: Subject J motion index

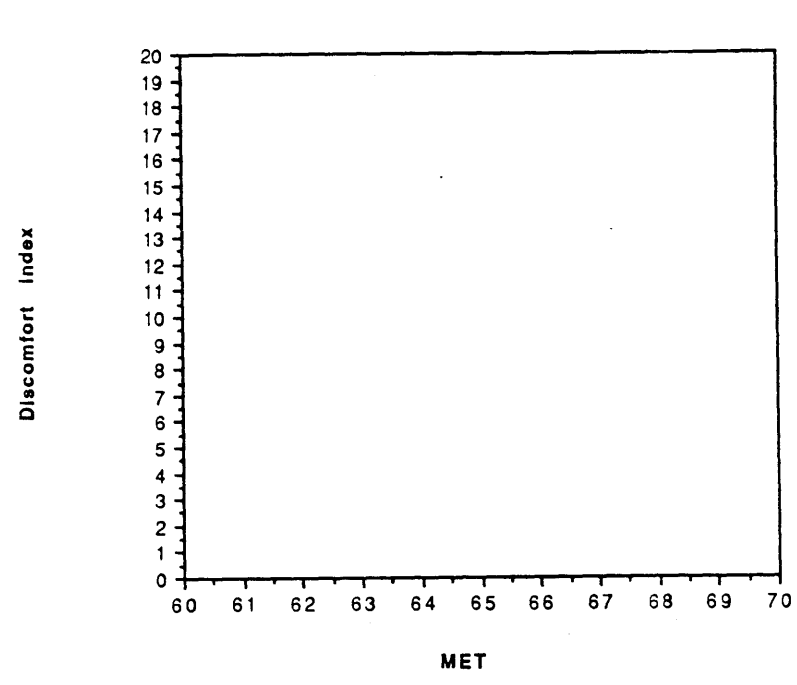

Figure 4.100: Subject J discomfort index

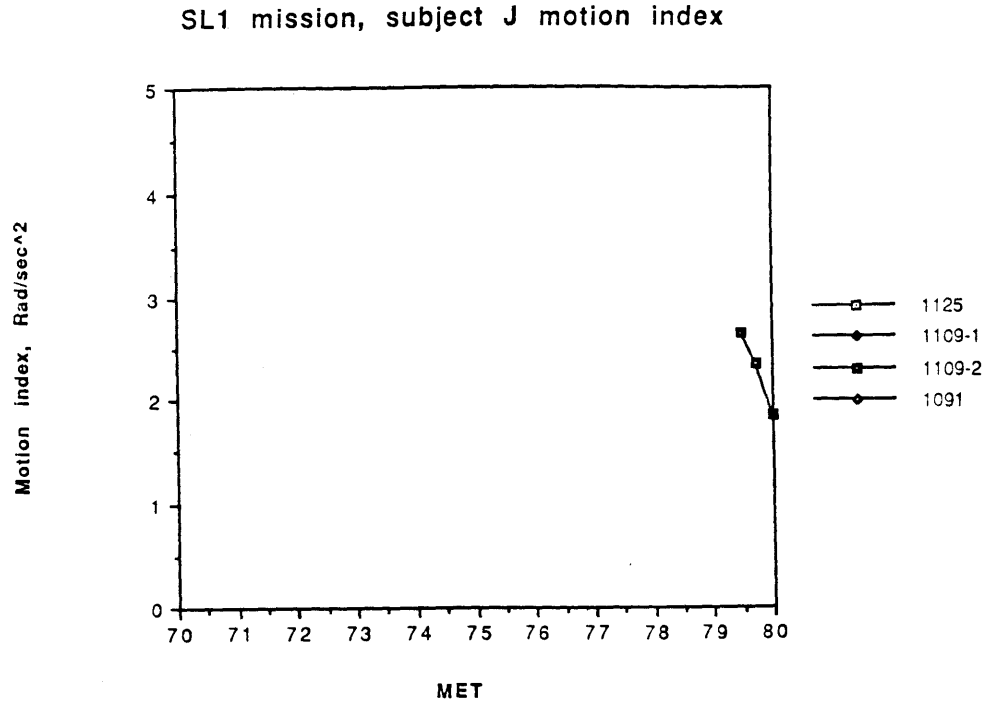

Figure 4.101: Subject J motion index

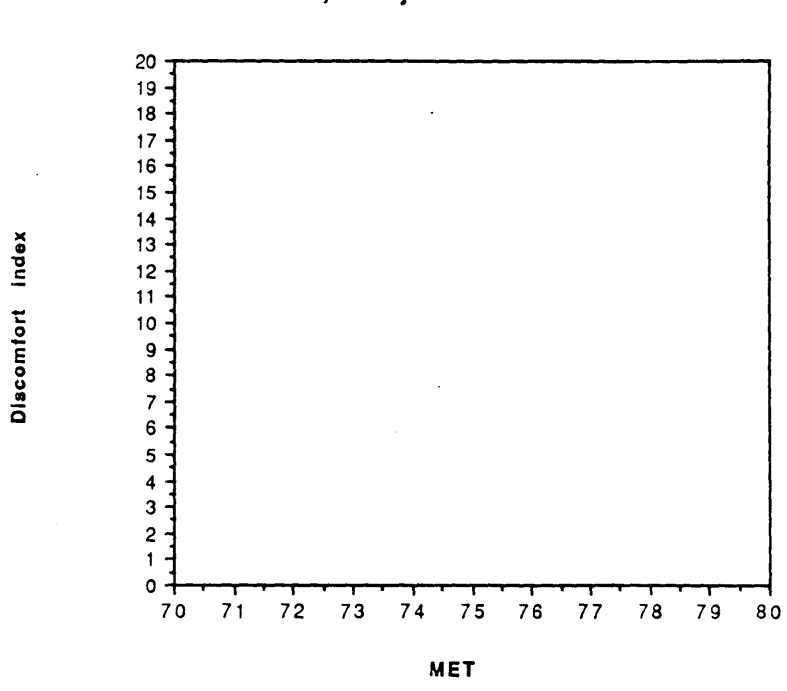

Figure 4.102: Subject J discomfort index

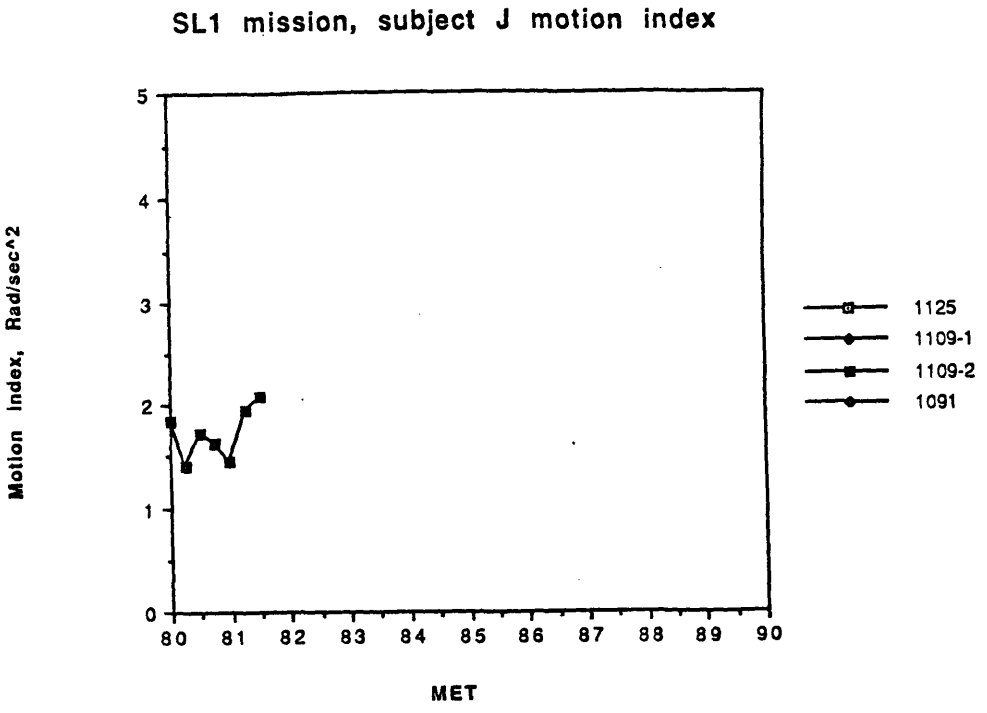

Figure 4.103: Subject J motion index

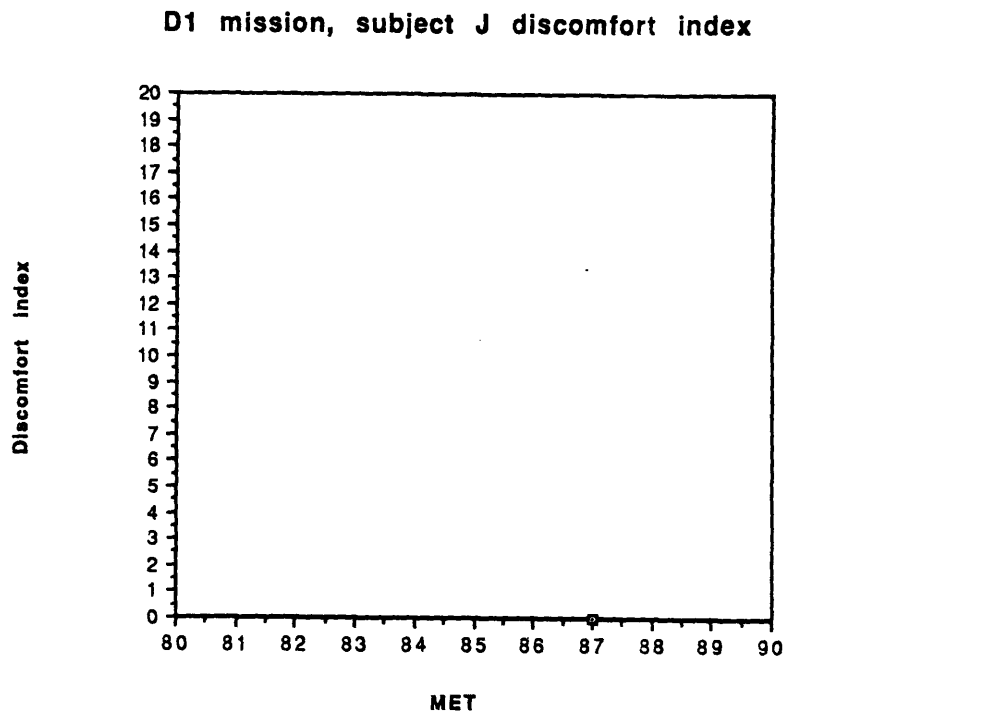

Figure 4.104: Subject J discomfort index

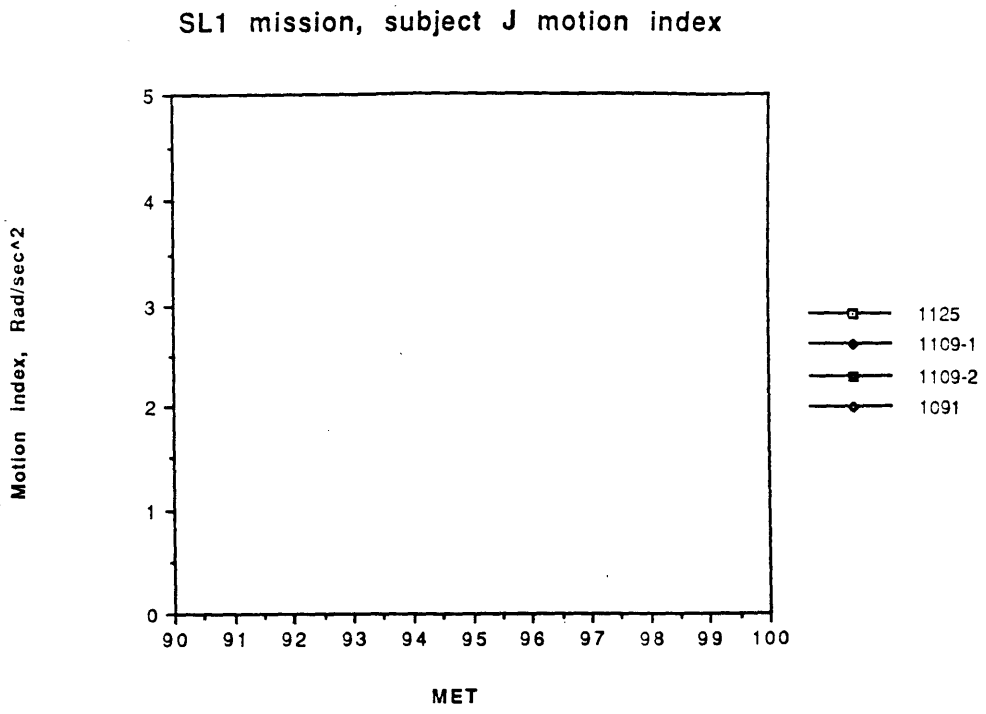

Figure 4.105: Subject J motion index

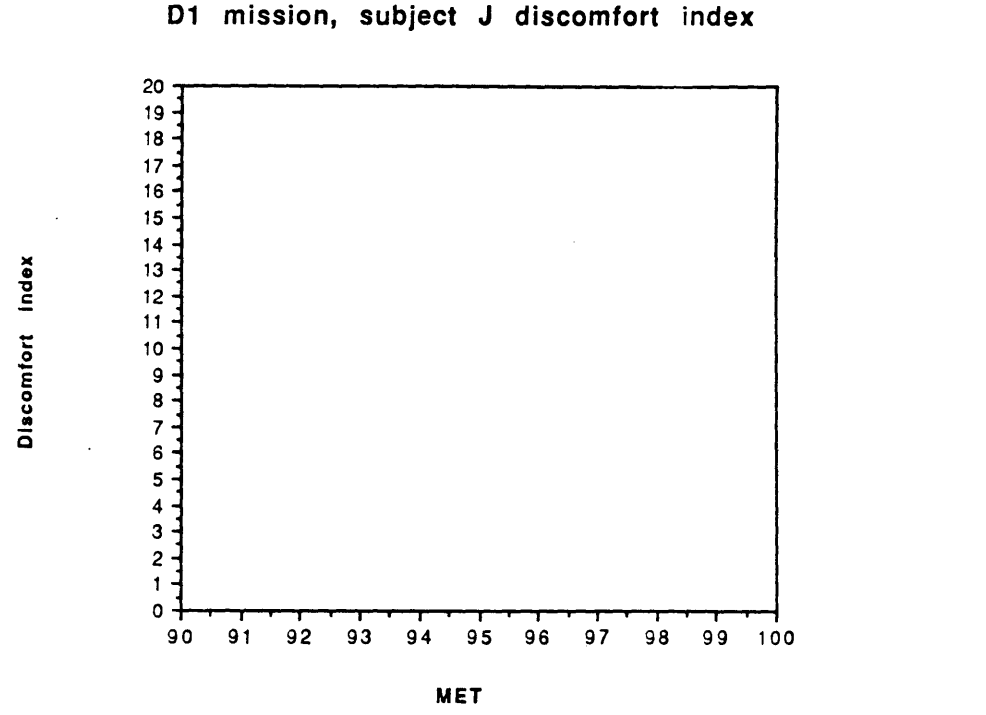

Figure 4.106: Subject J discomfort index

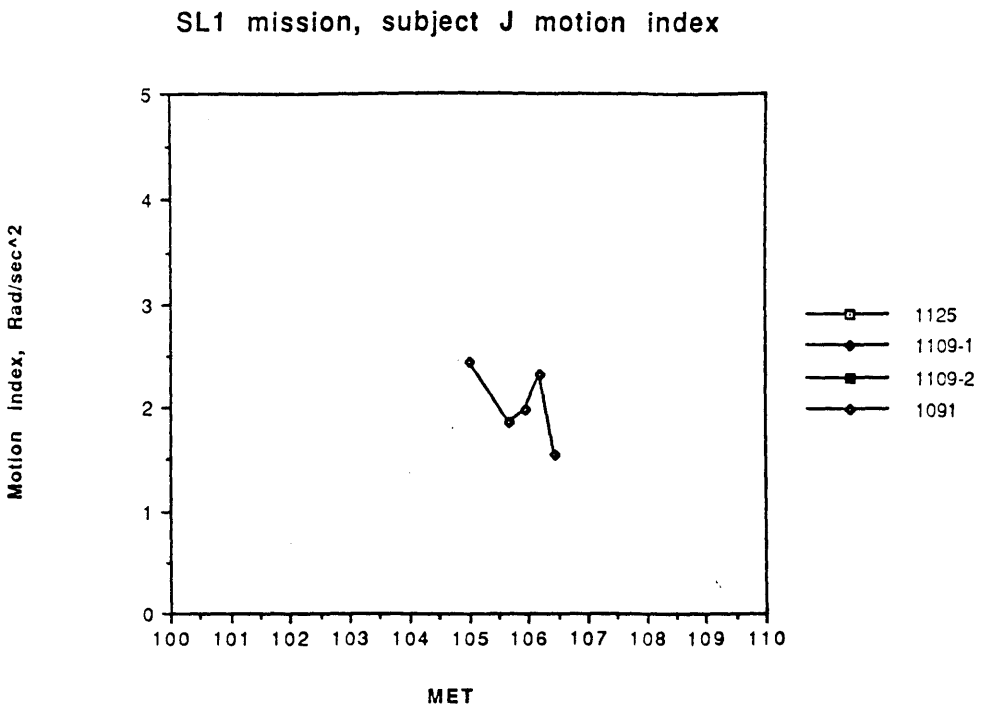

Figure 4.107: Subject J motion index

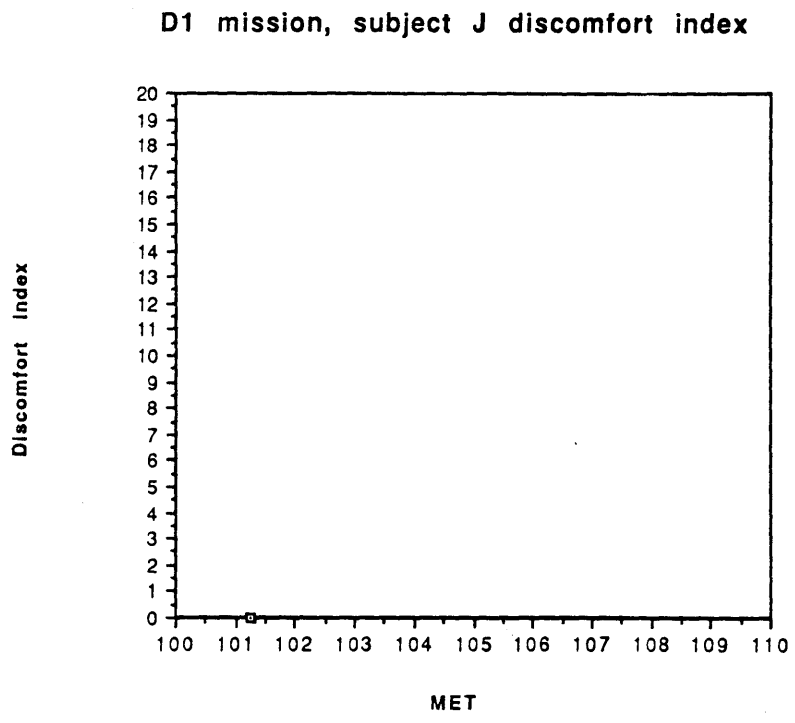

Figure 4.108: Subject J discomfort index

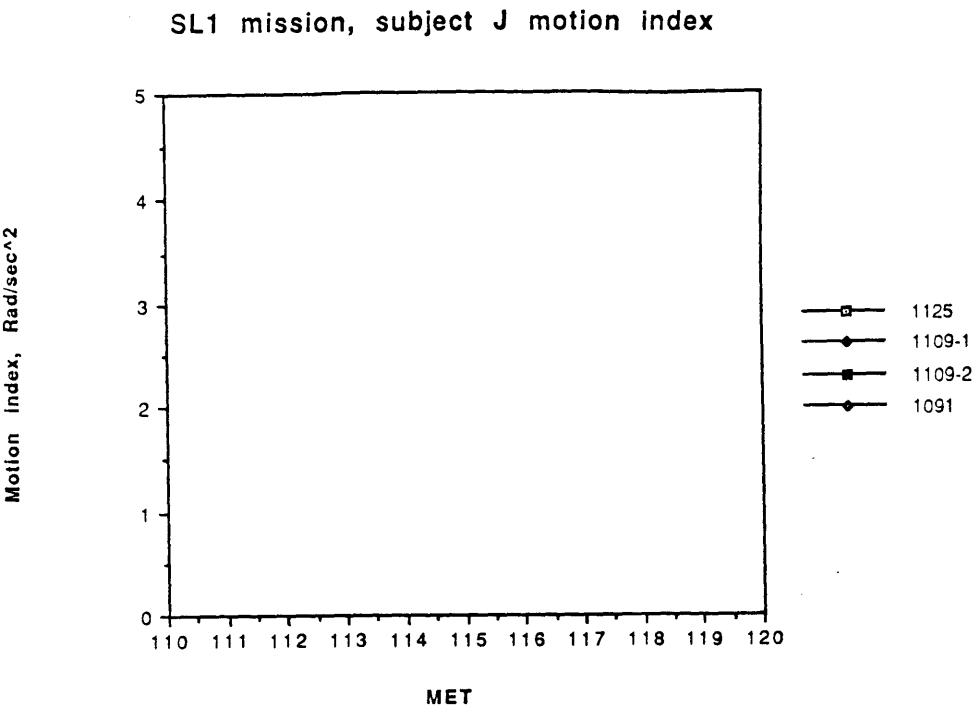

Figure 4.109: Subject J motion index

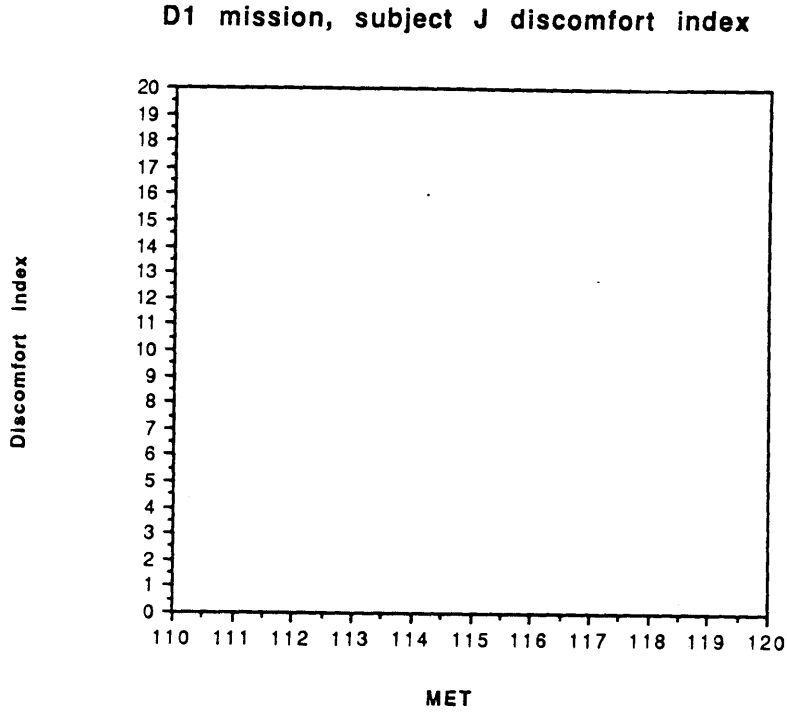

Figure 4.110: Subject J discomfort index

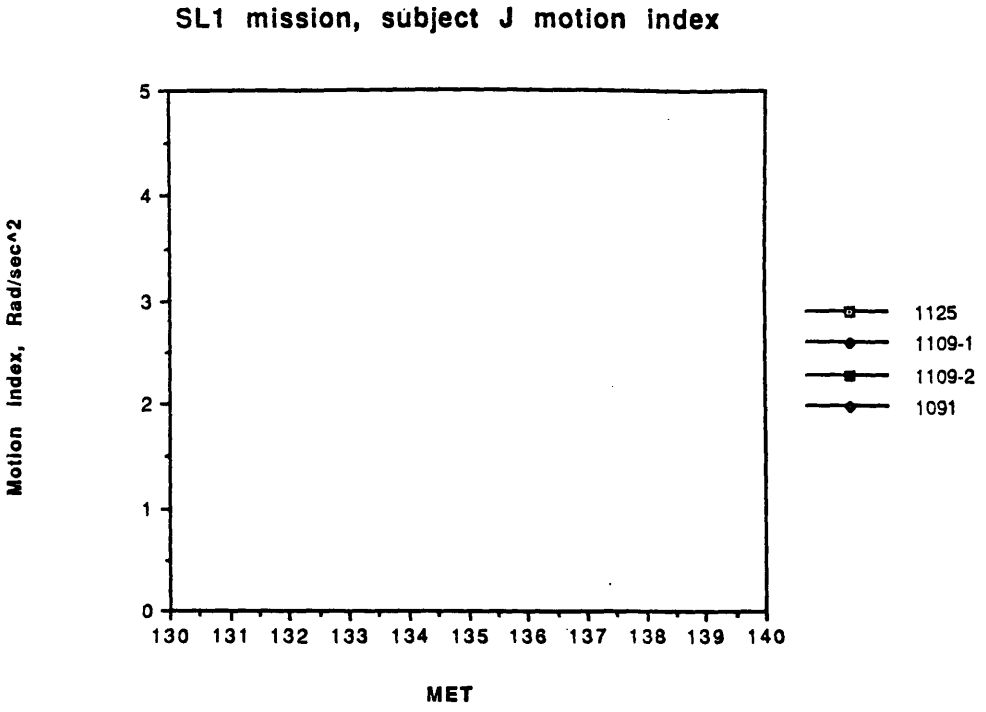

Figure 4.111: Subject J motion index

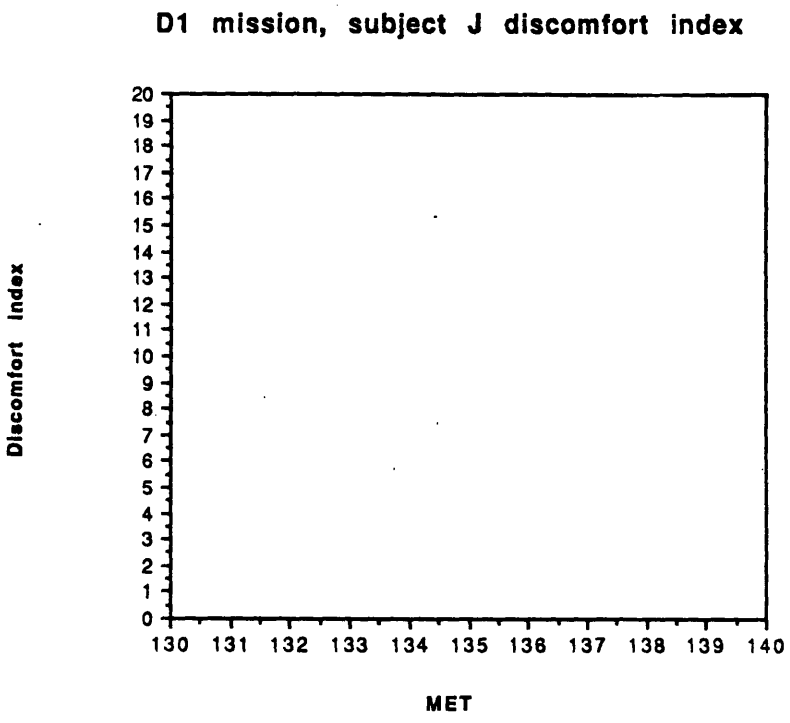

Figure 4.112: Subject J discomfort index

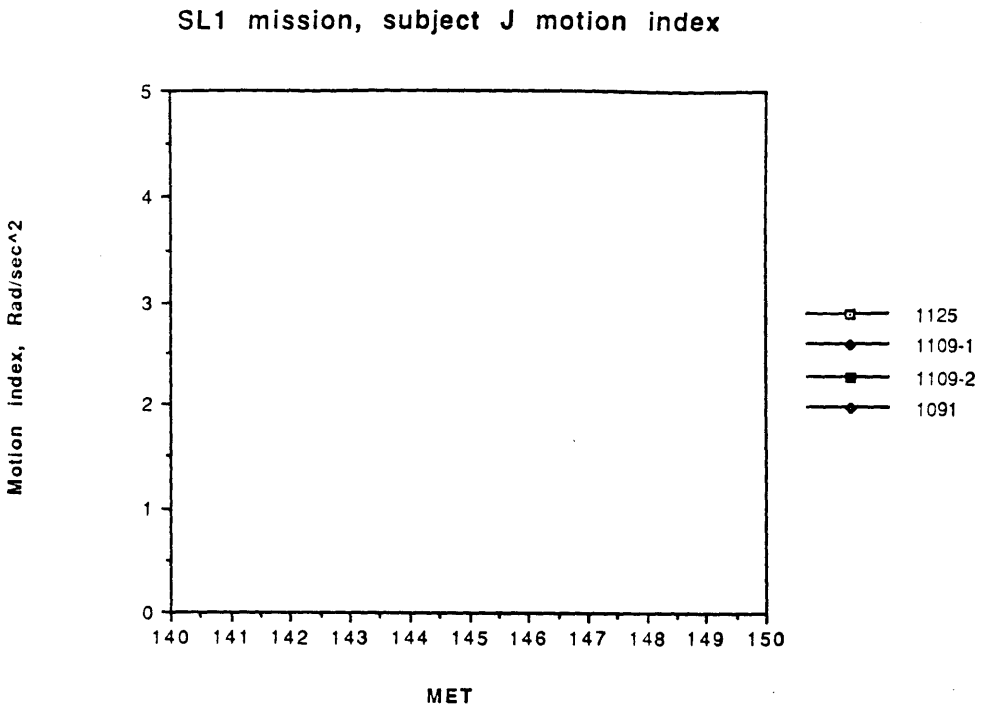

Figure 4.113: Subject J motion index

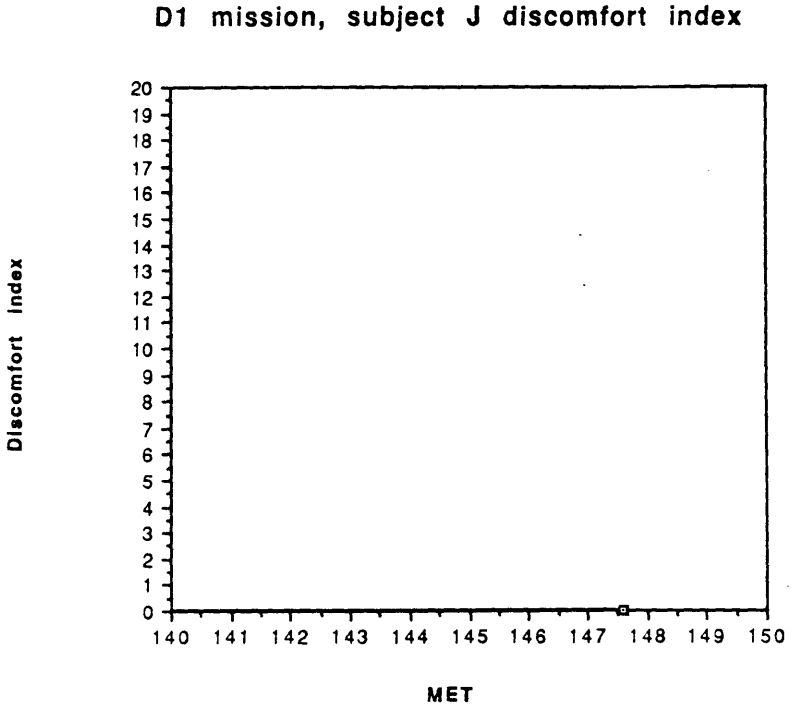

Figure 4.114: Subject J discomfort index

4.4 Results of Statistical Analysis - Head Motion and SMS Intensity

All four subjects were ranked for their SMS intensity. There are several criteria that could be used for this, for example the number of vomiting episodes, the average intensity while symptomatic, or how long the frank sickness lasted. All of these were similar. This ranking is, in increasing order, subject J, subject C, subject I and subject B, see figure:

Subject code	Ranking		
	Vomiting episodes	Average discomfort	Duration of frank sickness
B	4(6)	4	4(35 hours)
C	2(2)	2	2(14 hours)
I	3(3)	3	3(33 hours)
J	1(0)	1	1(-)

Figure 4.115: Subject SMS intensity ranking

All four subjects were ranked for their late mission motion index based on the RMS motion index results from the last 3 sessions. This ranking is not particularly sensitive to the exact number of sessions chosen. This ranking is, in increasing order, subject J, C, I, and B.

There appears to be a correlation between the astronauts head motion while asymptomatic and their SMS inflight sickness intensity. Subject B who has the highest motion index of the four subjects late in the mission also experienced the most severe SMS. Subject I had the next largest motion index late in the mission also had second in severety symptoms of SMS. Subject J did not experience SMS also had the lowest motion index. While it is hard to say anything definitively about subject C due to the lack of discomfort data, it does appear that he fits between subjects I and J in both head activity and discomfort.

4.4.1 Neck collar results

The neck collar experiment did not produce enough data to make statistical calculations. However, subject I did show a large decrease (from about 2.75 to about 1.5 Rad/sec^2) in the motion index. This is lower than his motion index while symptomatic. Therefore, it appears possible to use the neck collar as an anti-SMS device. Subject J did not show any difference in ARU data, which was also confirmed by a voice note reporting that he did not perceive any difference in his activity with or without the collar. We have no data on how tightly the collar was worn by both subjects.

4.5 Results of Statistical Analysis – Activity Indices vs. Discomfort Crossplots

Below are the cross plots of the motion index vs. discomfort index. This data is not very conclusive, but it is consistent with the hypothesis that astronauts tend to limit their head movements when they are symptomatic. Note that the majority of high motion indices are associated with low discomfort indices, as first noted in [8].

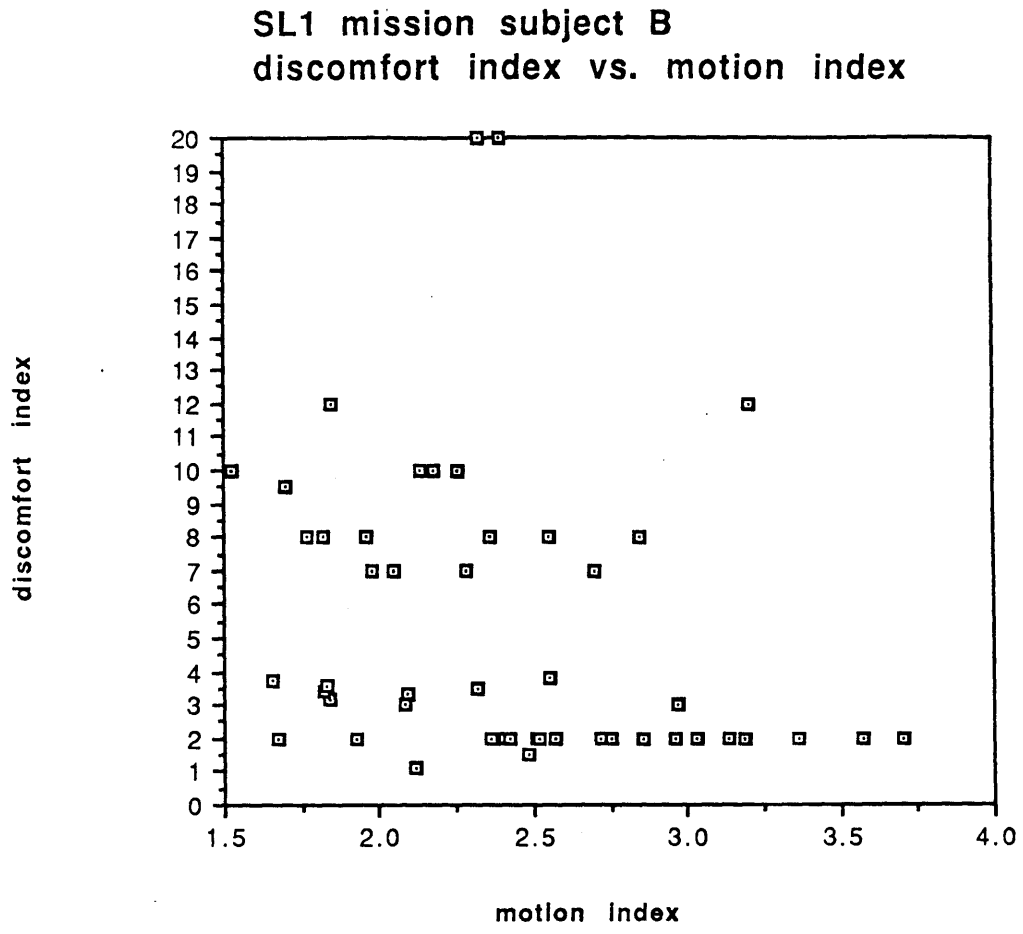

Figure 4.116: Motion index vs. discomfort index plot for subject B

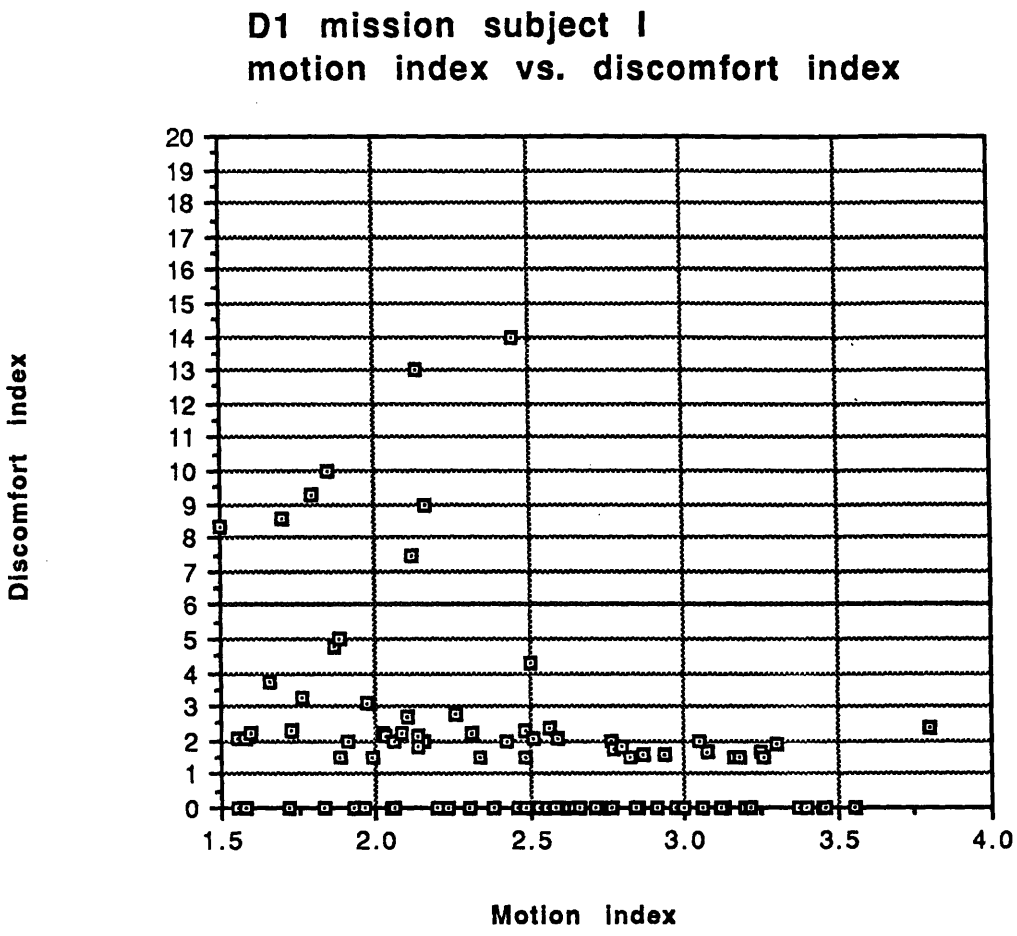

Figure 4.117: Motion index vs. discomfort index plot for subject I

Chapter 5

Conclusions and Suggestions for Further Research

5.1 Conclusions

Not enough ARU and discomfort data was taken during both missions to permit a complete statistical analysis. However, the data supports the hypotheses listed in previous chapter. There is a restriction of head motion, probably voluntary, for subject I during the period of increased SMS symptoms. The subject J who was hardly affected by SMS did not obviously change his head movement strategy. He was also less active than subject I when asymptomatic. Similar results were obtained for SL1 subjects B and C. Subject B strongly limited his head movement when symptomatic. Subject C did not show any clear trend over the mission. This supports the hypothesis that SMS intensity depends on the level of astronauts' natural head activity. In one of the two subjects who wore the neck collar, there was a dramatic decrease in head motion.

It appears possible to use the natural head activity level in weightlessness as a predictor of the SMS, or at least rank order of SMS intensity for different crewmembers. One could even speculate that astronauts are not likely to get sick if their head motion could be kept low, near 2 Rad/sec^2, and that they are very likely to get sick if it get high, near 3 Rad/sec^2. However, it is not believed that it is sufficient by itself. Rather

both it and the susceptibility to different visual and vestibular stimuli may have to be used (it is not clear, however, which of the stimuli should the astronauts be tested for).

5.2 Suggestions for Further Research

There are several suggestions for future research:

- For future missions, increase the amount of data taken, and ensure that there is good overlap of the discomfort data and the acceleration data. Switch to a more reliable recorder.
- Repeat the neck collar experiment. The neck collar should also be worn during the early part of a mission. If the motion index is reduced does discomfort index also improve?
- Test subjects on the ground to determine if head activity on earth is a good predictor of head activity in zero-g during late part of a mission
- Add 3 more linear accelerometers and place them symmetrically on two sides of the subject's head (keeping the angular accelerometers as well). This will permit easy separation of the part of linear acceleration data that's due to rotation. Then the comparison of rotational vs. translational stimuli can be made.

Appendix A

Programs

A.1 Description

There are several programs – ERROR_CLEANUP, AVERAGING, MANUAL_CLEAN, and R.K. McCoy's programs – PREPHG, DATAPREP. ERROR_CLEANUP is used for getting rid of the errors in the data files. It classifies all errors in a data file and then acts on them according to this classification. The program is written to make it easy to change what it does to each type of error — just change the parameters and recompile it. The program may clear error flags, set them, rescale data values, substitute a new value, or just report an error. It also reports all errors and error statistics.

AVERAGING is used for finding the averages of data on different channels. It only averages the data values whose error flags are cleared. The user can choose the channels and the intervals over which to average. The output files contain two column of numbers—the time and the average for that time. If there was not a single valid value in a specified region, the program closes the output file and opens a new one with a higher version number. That is done to signal to the user that there is missing data.

MANUAL_CLEAN is used for manually setting or clearing the error flag in the data

files. Large portions of data can be processed at a time. These three programs were written in Vax Fortran and compiled under VMS version 4.4 and 4.5. They cannot be compiled with a Fortran 77 compiler.

A.2 How To Use The Programs – ERROR_CLEANUP

To use ERROR_CLEANUP simply enter

```
$ RUN ERROR_CLEANUP
```

and reply to the prompts. First, the program will ask for the name of the data file to be processed. That is the raw data file. If the file is located on a different device, or in a different directory, just include the device and directory name in the file specifications. The complete file specifications have the form:

node::device:[directory]filename.filetype;version

For more information on file specs see the VAX VMS User's manual. Then the program will ask for the output file name. That is the name for the file containing the corrected data. The last file name that you will need to supply is for the comments and statistics. You will type it in when the program asks for report file name.

After you give the three file names, you'll be asked to give the first and last records in the data file to be processed. If the data file contains a text header record then the first record MUST be at least 4. If there is no header record, you may specify any number larger than 2 for the first record (The only way to get a file without a header is to use the CHOP program to cut a portion of a file out or process a file once with the ERROR_CLEANUP). The last record should be at least one record before the end of file. The records are 5120 bytes, or 2560 words long, except for the header record which has a different format. The data records contain the data for channels 1 through 8 repeated 320 times. At the sampling rate of 100Hz this allows storing of 3.2 seconds of data in each record. At the present, the program will work correctly even if you give the last record number that is larger than the last record in the file, but it

will not produce the final parts of a report file. Therefore you will not know if there were any flushed errors and will not have the statistics (see later for explanations). To determine how many records there are in a file give the DIR/SIZE command at the DCL prompt—$. That will show you the size of the file in blocks. Divide that number by 10 and you'll get the number of records it contains.

After that you'll be prompted for the channels to be processed. If you would like to process all channels enter 0, otherwise enter one number at a time, and enter -1 when done. This is the "official" channel assignment used in D1 experiments [1]:

Channel	Contents
1	X acceleration
2	Y acceleration
3	Z acceleration
4	Skin pallor
5	Roll acceleration
6	Skin temperature
7	Pitch acceleration
8	Yaw acceleration

The last query you need to answer is the expected average values in the channels at the beginning of the first record to be processed. You can determine those by using one of the other programs available, such as WDTOUT from JSC. That information is needed in case there is an error very close to the first processed values. In this case the program will not yet know what the average should be, so it needs your input. If you give the average that is not correct and an error starts in that channel before the average is recomputed, chances are all of the processing on that channel will be incorrect.

[1] in real data channels 4 and 6 were swapped

When the program stops running you will find, in addition to the original data file, the file with corrected data, and the report file. The report file will contain various statistics which you may disregard if you are only interested in getting the corrected data. There are, however a couple of things you should check in that file. First, closer to the end of it, right after the last error description, but before the statistics, you may see messages like FLUSHING OUT ERROR ... That means there were unprocessed errors in some channels when the program finished processing. Check when those errors started. If they are short you can disregard them. If they are long, you may need to take them into account when analyzing the data. In any case, you can set the error flag on that data manually with MANUAL_CLEAN program. Second, check for very long errors. This version of the program automatically sets the error bit on all values in that error. However, that takes a lot of time and some other users may change the parameters in the program to avoid that. Make sure that you have the version that marks all long errors if you need to avoid all unmarked errors.

If you need more than just the summary on the errors, you will find it in the report file. The information contained there consists of:

- the beginning, end, and channel number of certain errors
- flushed errors
- the full summary, contains number of errors in a block, total length of errors in a block, percentage of erroneous values in a segment and the same information about the corrected file for every 15 minute block
- the brief summary contains the same information for the entire file

A.3 How To Use The Programs — AVERAGE

To use AVERAGE just enter RUN AVERAGE at the $ prompt. First you will be asked for the data file specifications. That is the name of the file containing CDTR data. If this file is not in you directory, just use the entire file specifications. Next, you will be asked for the channels you want processed. Enter each channel, then enter -1 when done. If you want all channels just enter 0. Then you'll be asked for output file specifications for each channel to be processed. You can enter any legal name, but DO NOT enter version number. This program may need to open several files with the same name, so version numbers must not be restricted. If the data file contains errors that are long enough, this program will close the output file, and open a new one with the same name but higher version number. This way you will know that there are periods containing no valid data, rather than think that the data is a straight line. Next you'll be asked for the first and last record to process. The defaults are third and last records, respectively. The last thing you have to enter is the number of values to average over. The program will accept any valid integer higher than 1. Probably the best value is 6000–that will average over one minute intervals. The processing will start when the first time code is found.

A.4 How To Use The Programs — MANUAL_CLEAN

To use MANUAL_CLEAN just enter RUN MANUAL_CLEAN at the $ prompt and then respond to the questions.

A.5 How The Programs Work — AVERAGE

This is a very straight forward program. It simply computes averages of valid data. Only data whose error bit is cleared is used. The processing starts following the first time code found after the user specified record and continues until the user specified ending record. Program computes averages over user specified number of values, whether valid or invalid. If there isn't any valid data in some segment (so the average in that segment can't be computed) the program will close the output file, and open another one with the same name and higher version when the first valid value is encountered. The output files consist of two columns of numbers—time and the averages. Time is in decimal format, e.g. 12:15 would be 12.25.

A.6 How The Programs Work — MANUAL_CLEAN

This is also a very straight forward program. It simply sets or clears the error flag on some segment of data according to user instructions. No analysis of data beyond time code check is performed. Time codes are not affected by it.

Appendix B

Raw D1PREPHG Output

B.1 D1SN1091

B.1.1 Time Coverage

There are two different segments in this file. In the first one the timing starts with 2:27 and continues until 7:6. This start time corresponds to MET day 4, hour 9:23, subject J. In the second one it starts at 7:49 and continues until 8:33, which is the last time code in this file. Both error and estimate bits were set on all the time codes stating at 8:24. This corresponds to the second hop and drop late in the mission, and there is data for two subjects in this segment. This segment was not used in the analysis, because there was only one point per subject, and this was during hop and drop experiments.

B.1.2 Processed Data

Below is the summary of the first time segment in this file:

Record# 1 Start - End Times 2: 0 - 2:40 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	587	0.7	89413	-0.01	0.02		0.02 - 0.03
2	4215	4.7	85785	-0.19	0.03		0.03 - 0.03
3	2025	2.3	87975	-0.16	0.03		0.03 - 0.03
5	1031	1.1	88969	-0.74	2.06		2.05 - 2.09
7	1016	1.1	88984	-0.64	2.99		2.98 - 3.04
8	1974	2.2	88026	-0.53	2.16		2.14 - 2.23

Record# 2 Start - End Times 2:41 - 2:56 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	310	0.3	89690	-0.01	0.02		0.02 - 0.02
2	6808	7.6	83192	-0.19	0.02		0.02 - 0.03

3	3065	3.4	86935	-0.16	0.02		0.02 - 0.02
5	284	0.3	89716	-0.75	1.60		1.60 - 1.61
7	206	0.2	89794	-0.56	2.28		2.28 - 2.29
8	1604	1.8	88396	-0.56	1.66		1.65 - 1.71

Record# 3 Start - End Times 2:57 - 3:11 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	315	0.4	89685	0.00	0.02		0.02 - 0.02
2	4614	5.1	85386	-0.19	0.03		0.03 - 0.03
3	2016	2.2	87984	-0.16	0.02		0.02 - 0.02
5	250	0.3	89750	-0.74	1.70		1.70 - 1.71
7	228	0.3	89772	-0.55	2.28		2.27 - 2.28
8	1595	1.8	88405	-0.51	1.93		1.91 - 1.98

Record# 4 Start - End Times 3:12 - 3:26 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	301	0.3	89699	0.00	0.03		0.03 - 0.03
2	2832	3.1	87168	-0.19	0.03		0.03 - 0.04
3	1285	1.4	88715	-0.16	0.02		0.02 - 0.02
5	258	0.3	89742	-0.73	1.84		1.84 - 1.85
7	155	0.2	89845	-0.53	2.60		2.60 - 2.61
8	1639	1.8	88361	-0.44	2.48		2.46 - 2.55

Record# 5 Start - End Times 3:27 - 3:41 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	93	0.1	89907	-0.01	0.02		0.02 - 0.02
2	6387	7.1	83613	-0.19	0.02		0.02 - 0.02
3	19533	21.7	70467	-0.16	0.01		0.01 - 0.02
5	349	0.4	89651	-0.83	1.39		1.39 - 1.40
7	397	0.4	89603	-0.69	1.68		1.67 - 1.69
8	4177	4.6	85823	-0.57	1.58		1.55 - 1.59

Record# 6 Start - End Times 3:42 - 3: 0 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	4273	4.7	85727	-0.01	0.00		0.00 - 0.00
2	809	0.9	89191	-0.19	0.00		0.00 - 0.00
3	9565	10.6	80435	-0.16	0.00		0.00 - 0.00
5	224	0.2	89776	-0.87	0.06		0.06 - 0.06
7	43	0.0	89957	-0.77	0.14		0.14 - 0.14
8	5492	6.1	84508	-0.84	0.10		0.10 - 0.11

Record# 7 Start - End Times 3:57 - 4:12 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	386	0.4	89614	-0.01	0.00		0.00 - 0.00
2	5388	6.0	84612	-0.19	0.00		0.00 - 0.00
3	1542	1.7	88458	-0.16	0.00		0.00 - 0.00
5	159	0.2	89841	-0.87	0.05		0.05 - 0.05
7	44	0.0	89956	-0.80	0.13		0.13 - 0.13
8	44818	49.8	45182	-0.88	0.02		0.02 - 0.04

Record# 8 Start - End Times 4:13 - 4:27 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	83315	92.6	6685	-0.02	0.00		0.00 - 0.01
2	71	0.1	89929	-0.19	0.00		0.00 - 0.00
3	196	0.2	89804	-0.16	0.00		0.00 - 0.00
5	297	0.3	89703	-0.88	0.03		0.03 - 0.03
7	42	0.0	89958	-0.82	0.12		0.12 - 0.12
8	66494	73.9	23506	-0.90	0.07		0.04 - 0.13

Record# 9 Start - End Times 4:28 - 4:42 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	85835	95.4	4165	-0.02	0.01		0.00 - 0.02
2	4801	5.3	85199	-0.19	0.00		0.00 - 0.00
3	127	0.1	89873	-0.16	0.00		0.00 - 0.00
5	132	0.1	89868	-0.88	0.18		0.18 - 0.18
7	88	0.1	89912	-0.81	0.27		0.27 - 0.27
8	65915	73.2	24085	-0.92	0.36		0.19 - 0.65

Record# 10 Start - End Times 4:43 - 4:57 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	62180	69.1	27820	-0.02	0.01		0.00 - 0.01
2	18733	20.8	71267	-0.19	0.01		0.00 - 0.01

3	1228	1.4	88772	-0.16	0.00		0.00 - 0.00
5	1222	1.4	88778	-0.87	0.68		0.68 - 0.70
7	179	0.2	89821	-0.75	0.64		0.64 - 0.64
8	27282	30.3	62718	-0.72	0.62		0.52 - 0.86

Record# 11 Start - End Times 4:58 - 5:12 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	53443	59.4	36557	-0.01	0.00		0.00 - 0.01
2	68	0.1	89932	-0.19	0.00		0.00 - 0.00
3	15	0.0	89985	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.87	0.17		0.17 - 0.17
7	15	0.0	89985	-0.80	0.20		0.20 - 0.20
8	931	1.0	89069	-0.65	0.13		0.13 - 0.13

Record# 12 Start - End Times 5:13 - 5:27 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	39015	43.3	50985	-0.02	0.00		0.00 - 0.01
2	75	0.1	89925	-0.19	0.00		0.00 - 0.00
3	19	0.0	89981	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.87	0.21		0.21 - 0.21
7	19	0.0	89981	-0.79	0.22		0.22 - 0.22
8	728	0.8	89272	-0.68	0.14		0.14 - 0.14

Record# 13 Start - End Times 5:28 - 5:43 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	17423	19.4	72577	-0.02	0.00		0.00 - 0.00
2	88	0.1	89912	-0.19	0.00		0.00 - 0.00
3	20	0.0	89980	-0.16	0.00		0.00 - 0.00
5	16	0.0	89984	-0.87	0.19		0.19 - 0.19
7	50	0.1	89950	-0.79	0.22		0.22 - 0.22
8	2621	2.9	87379	-0.81	0.19		0.19 - 0.20

Record# 14 Start - End Times 5:44 - 5: 0 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	90000	100.0	0	0.00	0.00		5.00 - 5.00
2	1223	1.4	88777	-0.19	0.00		0.00 - 0.00
3	51	0.1	89949	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.88	0.03		0.03 - 0.03
7	19	0.0	89981	-0.80	0.13		0.13 - 0.13
8	21773	24.2	68227	-0.87	0.04		0.03 - 0.05

Record# 15 Start - End Times 5:59 - 6:13 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	90000	100.0	0	0.00	0.00		5.00 - 5.00
2	10361	11.5	79639	-0.19	0.00		0.00 - 0.01
3	26	0.0	89974	-0.16	0.00		0.00 - 0.00
5	19	0.0	89981	-0.88	0.05		0.05 - 0.05
7	15	0.0	89985	-0.80	0.14		0.14 - 0.14
8	40969	45.5	49031	-0.88	0.03		0.03 - 0.05

Record# 16 Start - End Times 6:14 - 6:28 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	90000	100.0	0	0.00	0.00		5.00 - 5.00
2	1006	1.1	88994	-0.19	0.00		0.00 - 0.00
3	192	0.2	89808	-0.16	0.00		0.00 - 0.00
5	2753	3.1	87247	-0.88	0.04		0.04 - 0.04
7	36	0.0	89964	-0.80	0.13		0.13 - 0.13
8	53218	59.1	36782	-0.89	0.05		0.03 - 0.09

Record# 17 Start - End Times 6:29 - 6:43 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	90000	100.0	0	0.00	0.00		5.00 - 5.00
2	76	0.1	89924	-0.19	0.00		0.00 - 0.00
3	53	0.1	89947	-0.16	0.00		0.00 - 0.00
5	19	0.0	89981	-0.88	0.03		0.03 - 0.03
7	19	0.0	89981	-0.80	0.13		0.13 - 0.13
8	60673	67.4	29327	-0.90	0.08		0.05 - 0.14

Record# 18 Start - End Times 6:44 - 6:58 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	88501	98.3	1499	-0.02	0.00		0.00 - 0.00
2	71	0.1	89929	-0.19	0.00		0.00 - 0.00

3	23	0.0	89977	-0.16	0.00		0.00 - 0.00
5	23	0.0	89977	-0.88	0.02		0.02 - 0.02
7	15	0.0	89985	-0.81	0.13		0.13 - 0.13
8	63898	71.0	26102	-0.94	0.12		0.06 - 0.21

Record# 19 Start - End Times 6:59 - 7: 6 Count = 42720

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	7064	16.5	35656	-0.02	0.00		0.00 - 0.00
2	360	0.8	42360	-0.19	0.00		0.00 - 0.01
3	328	0.8	42392	-0.16	0.00		0.00 - 0.00
5	328	0.8	42392	-0.88	0.13		0.13 - 0.13
7	328	0.8	42392	-0.86	0.16		0.16 - 0.16
8	31757	74.3	10963	-0.94	0.22		0.11 - 0.39

This is the summary of the second time segment in this file:

Record# 1 Start - End Times 7:50 - 8: 4 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2840	3.2	87160	-0.01	0.07		0.07 - 0.07
2	1901	2.1	88099	-0.19	0.06		0.06 - 0.06
3	7709	8.6	82291	-0.15	0.09		0.08 - 0.10
5	5043	5.6	84957	-0.82	3.64		3.54 - 3.94
7	6734	7.5	83266	-0.44	5.65		5.44 - 6.26
8	5021	5.6	84979	-0.51	3.83		3.72 - 4.14

Record# 2 Start - End Times 8: 5 - 8:19 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	5198	5.8	84802	0.00	0.07		0.07 - 0.07
2	2831	3.1	87169	-0.19	0.05		0.05 - 0.06
3	7539	8.4	82461	-0.16	0.07		0.07 - 0.08
5	5013	5.6	84987	-0.76	3.25		3.16 - 3.51
7	6485	7.2	83515	-0.60	4.53		4.36 - 4.99
8	4782	5.3	85218	-0.51	3.89		3.78 - 4.18

Record# 3 Start - End Times 8:20 - 8:33 Count = 86240

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	66334	76.9	19906	0.01	0.14		0.07 - 0.25
2	65828	76.3	20412	-0.19	0.06		0.03 - 0.10
3	69302	80.4	16938	-0.17	0.13		0.06 - 0.24
5	66261	76.8	19979	-0.67	3.43		1.65 - 6.24
7	66458	77.1	19782	-0.66	6.45		3.09 - 11.74
8	66280	76.9	19960	-0.52	3.73		1.80 - 6.79

B.2 D1SN1093

B.2.1 Time Coverage

Time coverage: start time 2:41 in rec.21, end time 10:34. The starting time corresponds to MET day 5, hour 9:37, subject I.

B.2.2 Processed Data

Record# 1 Start - End Times 2:42 - 2:56 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	109	0.1	89891	0.03	0.03		0.03 - 0.03
2	243	0.3	89757	0.10	0.04		0.04 - 0.04
3	204	0.2	89796	-0.08	0.02		0.02 - 0.02
5	321	0.4	89679	-3.62	2.32		2.32 - 2.33
7	774	0.9	89226	-2.50	3.23		3.21 - 3.27
8	238	0.3	89762	-4.55	3.35		3.34 - 3.36

Record# 2 Start - End Times 2:57 - 3:11 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	54	0.1	89946	0.03	0.02		0.02 - 0.02
2	87	0.1	89913	0.09	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.01		0.01 - 0.01
5	20	0.0	89980	-5.96	1.12		1.12 - 1.12
7	59	0.1	89941	-2.56	1.62		1.61 - 1.62
8	94	0.1	89906	-5.89	1.91		1.91 - 1.92

Record# 3 Start - End Times 3:12 - 3:26 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	24	0.0	89976	0.03	0.02		0.02 - 0.02
2	75	0.1	89925	0.09	0.02		0.02 - 0.02
3	59	0.1	89941	-0.08	0.01		0.01 - 0.01
5	126	0.1	89874	-6.94	1.02		1.01 - 1.02
7	76	0.1	89924	-2.58	1.70		1.70 - 1.70
8	118	0.1	89882	-6.39	1.81		1.81 - 1.81

Record# 4 Start - End Times 3:27 - 3:41 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	265	0.3	89735	0.03	0.03		0.03 - 0.03
2	233	0.3	89767	0.09	0.04		0.04 - 0.04
3	254	0.3	89746	-0.09	0.02		0.02 - 0.02
5	276	0.3	89724	-5.75	2.37		2.36 - 2.38
7	815	0.9	89185	-2.50	3.30		3.28 - 3.34
8	367	0.4	89633	-5.98	3.15		3.15 - 3.17

Record# 5 Start - End Times 3:42 - 3:56 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	101	0.1	89899	0.03	0.02		0.02 - 0.02
2	223	0.2	89777	0.09	0.03		0.03 - 0.03
3	212	0.2	89788	-0.09	0.02		0.02 - 0.02
5	108	0.1	89892	-2.88	1.90		1.90 - 1.90
7	364	0.4	89636	-2.42	3.29		3.29 - 3.31
8	234	0.3	89766	-3.79	2.82		2.82 - 2.83

Record# 6 Start - End Times 3:57 - 4:11 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	73	0.1	89927	0.02	0.03		0.03 - 0.03
2	122	0.1	89878	0.09	0.04		0.04 - 0.04
3	192	0.2	89808	-0.09	0.02		0.02 - 0.02
5	272	0.3	89728	-2.01	1.91		1.91 - 1.92
7	383	0.4	89617	-2.43	2.72		2.72 - 2.74
8	154	0.2	89846	-3.98	4.12		4.12 - 4.13

Record# 7 Start - End Times 4:12 - 4:26 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	58	0.1	89942	0.03	0.03		0.03 - 0.03
2	89	0.1	89911	0.09	0.04		0.04 - 0.04
3	78	0.1	89922	-0.09	0.02		0.02 - 0.02
5	49	0.1	89951	-1.91	1.53		1.53 - 1.53
7	192	0.2	89808	-2.45	2.22		2.21 - 2.22
8	108	0.1	89892	-4.19	3.47		3.47 - 3.48

Record# 8 Start - End Times 4:27 - 4:41 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.02		0.02 - 0.02
2	71	0.1	89929	0.09	0.03		0.03 - 0.03
3	20	0.0	89980	-0.08	0.02		0.02 - 0.02
5	56	0.1	89944	-1.55	1.23		1.23 - 1.23
7	45	0.1	89955	-2.44	1.93		1.93 - 1.93
8	80	0.1	89920	-4.65	3.12		3.12 - 3.13

Record# 9 Start - End Times 4:42 - 4:56 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	91	0.1	89909	0.03	0.03		0.03 - 0.03
2	95	0.1	89905	0.09	0.04		0.04 - 0.04
3	109	0.1	89891	-0.08	0.02		0.02 - 0.02
5	139	0.2	89861	-1.04	1.70		1.70 - 1.70
7	324	0.4	89676	-2.41	2.71		2.70 - 2.72
8	167	0.2	89833	-4.21	3.77		3.77 - 3.78

Record# 10 Start - End Times 4:57 - 5:12 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	88	0.1	89912	0.03	0.03		0.03 - 0.03
2	147	0.2	89853	0.09	0.04		0.04 - 0.04
3	196	0.2	89804	-0.09	0.02		0.02 - 0.02
5	146	0.2	89854	-2.05	1.72		1.71 - 1.72
7	430	0.5	89570	-2.46	2.77		2.76 - 2.79
8	137	0.2	89863	-3.80	3.44		3.44 - 3.45

Record# 11 Start - End Times 5:13 - 5:27 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	819	0.9	89181	0.03	0.03		0.03 - 0.03
2	319	0.4	89681	0.09	0.04		0.04 - 0.04
3	407	0.5	89593	-0.09	0.03		0.03 - 0.03
5	714	0.8	89286	-2.62	2.36		2.35 - 2.39
7	1584	1.8	88416	-2.45	3.35		3.32 - 3.44
8	807	0.9	89193	-3.12	3.56		3.54 - 3.60

Record# 12 Start - End Times 5:28 - 5:42 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	6946	7.7	83054	0.03	0.00		0.00 - 0.00
2	75	0.1	89925	0.09	0.01		0.01 - 0.01
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	39	0.0	89961	-1.65	0.19		0.19 - 0.19
7	61	0.1	89939	-2.58	0.11		0.11 - 0.11
8	87	0.1	89913	-4.55	0.18		0.18 - 0.18

Record# 13 Start - End Times 5:43 - 5:57 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	482	0.5	89518	0.03	0.00		0.00 - 0.00
2	102	0.1	89898	0.10	0.00		0.00 - 0.00
3	22	0.0	89978	-0.08	0.00		0.00 - 0.00
5	803	0.9	89197	-0.99	0.19		0.19 - 0.19
7	83	0.1	89917	-2.54	0.14		0.14 - 0.14
8	5612	6.2	84388	-4.03	0.23		0.23 - 0.25

Record# 14 Start - End Times 5:58 - 6:12 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	665	0.7	89335	0.03	0.00		0.00 - 0.00
2	102	0.1	89898	0.10	0.00		0.00 - 0.00
3	201	0.2	89799	-0.08	0.00		0.00 - 0.00
5	16677	18.5	73323	-0.56	0.13		0.12 - 0.16
7	153	0.2	89847	-2.51	0.14		0.14 - 0.14
8	4832	5.4	85168	-3.53	0.11		0.11 - 0.12

Record# 15 Start - End Times 6:13 - 6:27 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	67	0.1	89933	0.10	0.00		0.00 - 0.00
3	19	0.0	89981	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	-0.14	0.19		0.19 - 0.19
7	19	0.0	89981	-2.50	0.14		0.14 - 0.14
8	773	0.9	89227	-3.19	0.16		0.16 - 0.17

Record# 16 Start - End Times 6:28 - 6:42 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	208	0.2	89792	0.03	0.00		0.00 - 0.00
2	106	0.1	89894	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	1965	2.2	88035	0.81	0.27		0.27 - 0.28
7	15	0.0	89985	-2.43	0.14		0.14 - 0.14
8	76	0.1	89924	-2.52	0.26		0.26 - 0.26

Record# 17 Start - End Times 6:43 - 6:58 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	16	0.0	89984	0.03	0.00		0.00 - 0.00
2	170	0.2	89830	0.10	0.00		0.00 - 0.00
3	16	0.0	89984	-0.08	0.00		0.00 - 0.00
5	16	0.0	89984	1.50	0.23		0.23 - 0.23
7	16	0.0	89984	-2.38	0.10		0.10 - 0.10
8	4807	5.3	85193	-1.91	0.16		0.16 - 0.18

Record# 18 Start - End Times 6:59 - 7:13 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	75	0.1	89925	0.10	0.00		0.00 - 0.00
3	41	0.0	89959	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	2.06	0.16		0.16 - 0.16
7	15	0.0	89985	-2.35	0.08		0.08 - 0.08
8	79	0.1	89921	-1.46	0.13		0.13 - 0.13

Record# 19 Start - End Times 7:14 - 7:28 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	132	0.1	89868	0.03	0.00		0.00 - 0.00
2	356	0.4	89644	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	2.33	0.07		0.07 - 0.07
7	15	0.0	89985	-2.35	0.08		0.08 - 0.08
8	75	0.1	89925	-1.25	0.13		0.13 - 0.13

Record# 20 Start - End Times 7:29 - 7:43 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	870	1.0	89130	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	2.34	0.04		0.04 - 0.04
7	42	0.0	89958	-2.35	0.08		0.08 - 0.08
8	67	0.1	89933	-1.31	0.14		0.14 - 0.14

Record# 21 Start - End Times 7:44 - 7:58 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	104	0.1	89896	0.03	0.00		0.00 - 0.00
2	545	0.6	89455	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	2.35	0.08		0.08 - 0.08
7	15	0.0	89985	-2.35	0.09		0.09 - 0.09
8	75	0.1	89925	-1.38	0.13		0.13 - 0.13

Record# 22 Start - End Times 7:59 - 8:13 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	19	0.0	89981	0.03	0.00		0.00 - 0.00
2	71	0.1	89929	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	2.63	0.11		0.11 - 0.11
7	15	0.0	89985	-2.35	0.07		0.07 - 0.07
8	71	0.1	89929	-1.09	0.52		0.52 - 0.52

Record# 23 Start - End Times 8:14 - 8:28 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	75	0.1	89925	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	2.78	0.15		0.15 - 0.15
7	15	0.0	89985	-2.36	0.09		0.09 - 0.09
8	75	0.1	89925	-0.91	0.10		0.10 - 0.10

Record# 24 Start - End Times 8:29 - 8:43 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	75	0.1	89925	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	1.95	0.92		0.92 - 0.92
7	29	0.0	89971	-2.44	0.28		0.28 - 0.28
8	91	0.1	89909	-2.02	1.20		1.20 - 1.20

Record# 25 Start - End Times 8:44 - 8:59 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	20	0.0	89980	0.03	0.00		0.00 - 0.00
2	356	0.4	89644	0.10	0.00		0.00 - 0.00
3	16	0.0	89984	-0.08	0.00		0.00 - 0.00
5	16	0.0	89984	0.68	0.16		0.16 - 0.16
7	20	0.0	89980	-2.55	0.18		0.18 - 0.18
8	80	0.1	89920	-3.52	0.08		0.08 - 0.08

Record# 26 Start - End Times 9: 0 - 9:14 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	811	0.9	89189	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	0.64	0.12		0.12 - 0.12
7	15	0.0	89985	-2.55	0.13		0.13 - 0.13
8	716	0.8	89284	-3.51	0.09		0.09 - 0.09

Record# 27 Start - End Times 9:15 - 9:29 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	75	0.1	89925	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	0.87	0.03		0.03 - 0.03
7	15	0.0	89985	-2.52	0.15		0.15 - 0.15
8	1445	1.6	88555	-3.32	0.13		0.13 - 0.13

Record# 28 Start - End Times 9:30 - 9:44 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.00		0.00 - 0.00
2	208	0.2	89792	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	15	0.0	89985	0.65	0.15		0.15 - 0.15
7	15	0.0	89985	-2.51	0.14		0.14 - 0.14
8	7430	8.3	82570	-3.55	0.13		0.12 - 0.15

Record# 29 Start - End Times 9:45 - 9:59 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	25	0.0	89975	0.03	0.00		0.00 - 0.00
2	75	0.1	89925	0.10	0.00		0.00 - 0.00
3	15	0.0	89985	-0.08	0.00		0.00 - 0.00
5	20	0.0	89980	0.22	0.21		0.21 - 0.21
7	15	0.0	89985	-2.53	0.14		0.14 - 0.14
8	3598	4.0	86402	-3.81	0.09		0.09 - 0.10

Record# 30 Start - End Times 10: 0 - 10:14 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	1464	1.6	88536	0.03	0.00		0.00 - 0.00
2	428	0.5	89572	0.10	0.00		0.00 - 0.00
3	116	0.1	89884	-0.08	0.00		0.00 - 0.00
5	4083	4.5	85917	0.05	0.12		0.12 - 0.13
7	29	0.0	89971	-2.53	0.17		0.17 - 0.17
8	24542	27.3	65458	-3.84	0.11		0.09 - 0.15

Record# 31 Start - End Times 10:15 - 10:29 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	3325	3.7	86675	0.03	0.00		0.00 - 0.00
2	2580	2.9	87420	0.10	0.00		0.00 - 0.00
3	369	0.4	89631	-0.08	0.00		0.00 - 0.00
5	16203	18.0	73797	0.47	0.20		0.18 - 0.25
7	89	0.1	89911	-2.51	0.23		0.23 - 0.23
8	15228	16.9	74772	-3.61	0.21		0.19 - 0.25

Record# 32 Start - End Times 10:30 - 10:34 Count = 28240

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	118	0.4	28122	0.03	0.00		0.00 - 0.00
2	900	3.2	27340	0.10	0.00		0.00 - 0.00
3	31	0.1	28209	-0.08	0.00		0.00 - 0.00
5	21939	77.7	6301	0.86	0.07		0.03 - 0.12
7	30	0.1	28210	-2.49	0.20		0.20 - 0.20
8	10279	36.4	17961	-3.36	0.14		0.11 - 0.21

B.3 D1SN1094

B.3.1 Time Coverage

The first time stamp in this file is 12:12. The last time stamp is 6:56. This starting time corresponds to MET day 1, hour 19:20, subject I.

B.3.2 Processed Data

Record# 1 Start - End Times 12:21 - 12:35 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	351	0.4	89649	0.03	0.02		0.02 - 0.02
2	187	0.2	89813	0.11	0.03		0.03 - 0.03
3	113	0.1	89887	-0.08	0.02		0.02 - 0.02
5	263	0.3	89737	-3.58	1.76		1.76 - 1.77
7	360	0.4	89640	-2.58	2.90		2.89 - 2.91
8	237	0.3	89763	-5.66	2.85		2.84 - 2.86
Record# 2 Start - End Times 12:36 - 12:50 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	1795	2.0	88205	0.03	0.04		0.04 - 0.04
2	564	0.6	89436	0.10	0.04		0.04 - 0.04
3	988	1.1	89012	-0.08	0.04		0.04 - 0.04
5	1890	2.1	88110	-3.22	3.45		3.41 - 3.55
7	8944	9.9	81056	-2.54	4.17		3.96 - 4.75
8	3304	3.7	86696	-4.83	3.77		3.70 - 3.98
Record# 3 Start - End Times 12:51 - 1: 5 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	60	0.1	89940	0.03	0.02		0.02 - 0.02
2	99	0.1	89901	0.10	0.03		0.03 - 0.03
3	31	0.0	89969	-0.09	0.02		0.02 - 0.02
5	65	0.1	89935	-2.25	1.57		1.57 - 1.57
7	353	0.4	89647	-2.50	3.15		3.14 - 3.16
8	146	0.2	89854	-3.46	2.47		2.47 - 2.48
Record# 4 Start - End Times 1: 6 - 1:20 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	102	0.1	89898	0.03	0.02		0.02 - 0.02
2	116	0.1	89884	0.10	0.03		0.03 - 0.03
3	57	0.1	89943	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-2.16	1.40		1.40 - 1.40
7	105	0.1	89895	-2.49	2.59		2.58 - 2.59
8	91	0.1	89909	-3.31	2.09		2.08 - 2.09
Record# 5 Start - End Times 1:21 - 1:35 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	24	0.0	89976	0.03	0.02		0.02 - 0.02
2	79	0.1	89921	0.10	0.03		0.03 - 0.03
3	22	0.0	89978	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-1.63	1.42		1.42 - 1.42
7	51	0.1	89949	-2.46	2.89		2.89 - 2.89
8	71	0.1	89929	-2.90	2.38		2.38 - 2.39
Record# 6 Start - End Times 1:36 - 1:50 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	23	0.0	89977	0.03	0.02		0.02 - 0.02
2	91	0.1	89909	0.10	0.03		0.03 - 0.03
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	27	0.0	89973	-1.95	1.35		1.35 - 1.35
7	78	0.1	89922	-2.48	2.62		2.62 - 2.62
8	133	0.1	89867	-3.23	2.24		2.23 - 2.24
Record# 7 Start - End Times 1:51 - 2: 5 Count = 90000							
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	40	0.0	89960	0.03	0.02		0.02 - 0.02
2	182	0.2	89818	0.10	0.03		0.03 - 0.03
3	54	0.1	89946	-0.08	0.02		0.02 - 0.02
5	51	0.1	89949	-1.96	1.59		1.59 - 1.59
7	120	0.1	89880	-2.45	2.81		2.81 - 2.82
8	126	0.1	89874	-3.16	3.11		3.11 - 3.12

Record# 8 Start - End Times 2: 6 - 2:20 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	19	0.0	89981	0.03	0.02		0.02 - 0.02
2	79	0.1	89921	0.10	0.03		0.03 - 0.03
3	32	0.0	89968	-0.08	0.02		0.02 - 0.02
5	46	0.1	89954	-1.45	1.60		1.60 - 1.61
7	83	0.1	89917	-2.43	3.25		3.25 - 3.26
8	118	0.1	89882	-2.72	2.41		2.41 - 2.41

Record# 9 Start - End Times 2:21 - 2:35 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	59	0.1	89941	0.03	0.02		0.02 - 0.02
2	95	0.1	89905	0.10	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-1.22	1.38		1.38 - 1.38
7	46	0.1	89954	-2.43	2.61		2.61 - 2.61
8	94	0.1	89906	-2.55	1.99		1.99 - 1.99

Record# 10 Start - End Times 2:36 - 2:50 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	101	0.1	89899	0.03	0.03		0.03 - 0.03
2	152	0.2	89848	0.10	0.04		0.04 - 0.04
3	33	0.0	89967	-0.08	0.03		0.03 - 0.03
5	80	0.1	89920	-1.14	1.95		1.94 - 1.95
7	304	0.3	89696	-2.39	3.97		3.96 - 3.99
8	271	0.3	89729	-2.45	2.89		2.88 - 2.90

Record# 11 Start - End Times 2:51 - 3: 5 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	121	0.1	89879	0.03	0.03		0.03 - 0.03
2	144	0.2	89856	0.10	0.03		0.03 - 0.03
3	80	0.1	89920	-0.08	0.03		0.03 - 0.03
5	132	0.1	89868	-0.93	2.15		2.15 - 2.15
7	905	1.0	89095	-2.40	4.48		4.45 - 4.54
8	297	0.3	89703	-2.31	2.85		2.84 - 2.86

Record# 12 Start - End Times 3: 6 - 3:20 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	61	0.1	89939	0.03	0.02		0.02 - 0.02
2	95	0.1	89905	0.10	0.03		0.03 - 0.03
3	68	0.1	89932	-0.08	0.02		0.02 - 0.02
5	127	0.1	89873	-1.17	1.88		1.88 - 1.88
7	856	1.0	89144	-2.47	3.57		3.56 - 3.63
8	229	0.3	89771	-2.65	2.71		2.70 - 2.72

Record# 13 Start - End Times 3:21 - 3:35 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	157	0.2	89843	0.03	0.02		0.02 - 0.02
2	172	0.2	89828	0.10	0.02		0.02 - 0.02
3	67	0.1	89933	-0.08	0.02		0.02 - 0.02
5	152	0.2	89848	-1.45	1.40		1.40 - 1.41
7	478	0.5	89522	-2.52	2.72		2.72 - 2.75
8	169	0.2	89831	-2.91	2.09		2.08 - 2.09

Record# 14 Start - End Times 3:36 - 3:50 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	213	0.2	89787	0.03	0.02		0.02 - 0.02
2	246	0.3	89754	0.10	0.03		0.03 - 0.03
3	108	0.1	89892	-0.08	0.02		0.02 - 0.02
5	398	0.4	89602	-1.60	1.96		1.96 - 1.98
7	793	0.9	89207	-2.50	3.42		3.40 - 3.46
8	449	0.5	89551	-2.61	2.75		2.74 - 2.77

Record# 15 Start - End Times 3:51 - 4: 5 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	45	0.1	89955	0.03	0.02		0.02 - 0.02
2	99	0.1	89901	0.10	0.04		0.04 - 0.04
3	26	0.0	89974	-0.08	0.03		0.03 - 0.03
5	86	0.1	89914	0.20	2.10		2.10 - 2.11
7	360	0.4	89640	-2.38	3.76		3.75 - 3.78
8	269	0.3	89731	-3.92	3.14		3.14 - 3.16

Record# 16 Start - End Times 4: 6 - 4:20 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	63	0.1	89937	0.03	0.02		0.02 - 0.02
2	105	0.1	89895	0.10	0.04		0.04 - 0.04
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	48	0.1	89952	0.03	2.29		2.29 - 2.29
7	222	0.2	89778	-2.38	3.59		3.58 - 3.60
8	335	0.4	89665	-4.01	3.68		3.68 - 3.70

Record# 17 Start - End Times 4:21 - 4:35 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	39	0.0	89961	0.03	0.02		0.02 - 0.02
2	72	0.1	89928	0.10	0.03		0.03 - 0.03
3	34	0.0	89966	-0.08	0.02		0.02 - 0.02
5	36	0.0	89964	0.13	2.04		2.04 - 2.04
7	265	0.3	89735	-2.39	3.42		3.42 - 3.44
8	289	0.3	89711	-3.90	2.97		2.97 - 2.99

Record# 18 Start - End Times 4:36 - 4:50 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	55	0.1	89945	0.03	0.02		0.02 - 0.02
2	115	0.1	89885	0.10	0.04		0.04 - 0.04
3	36	0.0	89964	-0.08	0.02		0.02 - 0.02
5	133	0.1	89867	0.36	2.36		2.36 - 2.37
7	281	0.3	89719	-2.39	3.41		3.41 - 3.43
8	340	0.4	89660	-3.66	2.97		2.96 - 2.98

Record# 19 Start - End Times 4:51 - 5: 6 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	141	0.2	89859	0.03	0.01		0.01 - 0.01
2	76	0.1	89924	0.10	0.03		0.03 - 0.03
3	16	0.0	89984	-0.08	0.02		0.02 - 0.02
5	16	0.0	89984	0.58	1.52		1.52 - 1.52
7	96	0.1	89904	-2.41	2.21		2.21 - 2.21
8	133	0.1	89867	-3.44	2.17		2.17 - 2.17

Record# 20 Start - End Times 5: 7 - 5:21 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	52	0.1	89948	0.02	0.02		0.02 - 0.02
2	113	0.1	89887	0.10	0.04		0.04 - 0.04
3	34	0.0	89966	-0.09	0.03		0.03 - 0.03
5	79	0.1	89921	-0.07	2.49		2.49 - 2.49
7	305	0.3	89695	-2.40	3.77		3.77 - 3.79
8	235	0.3	89765	-3.53	3.10		3.09 - 3.11

Record# 21 Start - End Times 5:22 - 5:36 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	38	0.0	89962	0.03	0.02		0.02 - 0.02
2	112	0.1	89888	0.10	0.03		0.03 - 0.03
3	51	0.1	89949	-0.09	0.02		0.02 - 0.02
5	129	0.1	89871	-4.11	2.01		2.01 - 2.01
7	418	0.5	89582	-2.51	2.72		2.71 - 2.74
8	195	0.2	89805	-3.40	2.22		2.22 - 2.23

Record# 22 Start - End Times 5:37 - 5:51 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	84	0.1	89916	0.02	0.03		0.02 - 0.03
2	141	0.2	89859	0.10	0.04		0.04 - 0.04
3	20	0.0	89980	-0.09	0.02		0.02 - 0.02
5	69	0.1	89931	-4.80	2.21		2.21 - 2.21
7	451	0.5	89549	-2.43	3.83		3.82 - 3.86
8	211	0.2	89789	-3.00	3.28		3.27 - 3.29

Record# 23 Start - End Times 5:52 - 6: 6 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	100	0.1	89900	0.03	0.01		0.01 - 0.01
2	160	0.2	89840	0.10	0.02		0.02 - 0.02
3	64	0.1	89936	-0.09	0.02		0.02 - 0.02
5	132	0.1	89868	-4.80	1.43		1.43 - 1.44
7	230	0.3	89770	-2.49	2.42		2.42 - 2.43
8	163	0.2	89837	-3.37	1.65		1.65 - 1.65

Record# 24 Start - End Times 6: 7 - 6:21 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	54	0.1	89946	0.02	0.02		0.02 - 0.02
2	85	0.1	89915	0.10	0.03		0.03 - 0.03
3	15	0.0	89985	-0.09	0.02		0.02 - 0.02
5	127	0.1	89873	-4.58	1.82		1.82 - 1.82
7	299	0.3	89701	-2.44	2.61		2.61 - 2.62
8	126	0.1	89874	-3.99	2.89		2.88 - 2.89

Record# 25 Start - End Times 6:22 - 6:36 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	115	0.1	89885	0.02	0.02		0.02 - 0.02
2	114	0.1	89886	0.10	0.04		0.04 - 0.04
3	27	0.0	89973	-0.09	0.02		0.02 - 0.02
5	253	0.3	89747	-4.42	2.47		2.47 - 2.48
7	978	1.1	89022	-2.39	3.86		3.83 - 3.92
8	386	0.4	89614	-3.38	3.31		3.30 - 3.33

Record# 26 Start - End Times 6:37 - 6:50 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	33526	37.3	56474	0.02	0.02		0.02 - 0.03
2	33591	37.3	56409	0.10	0.03		0.02 - 0.04
3	33353	37.1	56647	-0.09	0.02		0.02 - 0.03
5	33819	37.6	56181	-3.48	2.18		1.72 - 3.18
7	33921	37.7	56079	-2.41	3.39		2.68 - 4.95
8	33463	37.2	56537	-2.61	2.76		2.19 - 4.02

Record# 27 Start - End Times 6:51 - 6:56 Count = 32160

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	32160	100.0	0	0.00	0.00		5.00 - 5.00
2	32160	100.0	0	0.00	0.00		5.00 - 5.00
3	32160	100.0	0	0.00	0.00		5.00 - 5.00
5	32160	100.0	0	0.00	0.00		150.00 - 150.00
7	32160	100.0	0	0.00	0.00		150.00 - 150.00
8	32160	100.0	0	0.00	0.00		150.00 - 150.00

B.4 D1SN1109

B.4.1 Time Coverage

There are three segments in this file. The first one starts at 4:59 and continues until 10:12. This starting time corresponds to MET day 0, hour 23:56, subject J. The second one starts at 12:27 (around rec 5827) and continues until 2:40. This starting time corresponds to MET day 3, hour 7:28, subject J. The third one starts at 10:14 (in rec 8287). The last time stamp in the file is 10:35. It is not clear what this corresponds to. This segment was not used in the analysis.

B.4.2 Processed Data

Below is the summary from the first segment:

Record# 1 Start - End Times 4:59 - 12: 0 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	809	0.9	89191	-0.02	0.03		0.03 - 0.03
2	997	1.1	89003	-0.19	0.03		0.03 - 0.03
3	1417	1.6	88583	-0.16	0.03		0.03 - 0.03
5	2960	3.3	87040	-0.88	2.87		2.83 - 3.01
7	3093	3.4	86907	-0.74	3.20		3.15 - 3.36
8	7989	8.9	82011	-0.61	3.46		3.31 - 3.90

Record# 2 Start - End Times 5:14 - 5:28 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	231	0.3	89769	-0.02	0.02		0.02 - 0.02
2	389	0.4	89611	-0.19	0.03		0.03 - 0.03
3	600	0.7	89400	-0.16	0.02		0.02 - 0.02
5	302	0.3	89698	-0.85	1.56		1.55 - 1.56
7	210	0.2	89790	-0.73	1.96		1.96 - 1.97
8	2757	3.1	87243	-0.41	1.97		1.94 - 2.06

Record# 3 Start - End Times 5:29 - 5:43 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	115	0.1	89885	-0.01	0.02		0.02 - 0.02
2	918	1.0	89082	-0.19	0.02		0.02 - 0.02
3	800	0.9	89200	-0.16	0.02		0.02 - 0.02
5	104	0.1	89896	-0.83	1.39		1.39 - 1.40
7	31	0.0	89969	-0.71	1.75		1.75 - 1.75
8	2916	3.2	87084	-0.35	1.61		1.58 - 1.69

Record# 4 Start - End Times 5:44 - 5:59 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	62	0.1	89938	-0.01	0.02		0.02 - 0.02
2	1223	1.4	88777	-0.19	0.02		0.02 - 0.02
3	1065	1.2	88935	-0.16	0.02		0.02 - 0.02
5	155	0.2	89845	-0.81	1.33		1.33 - 1.34
7	77	0.1	89923	-0.65	1.77		1.77 - 1.77
8	2585	2.9	87415	-0.32	1.87		1.84 - 1.95

Record# 5 Start - End Times 6: 0 - 6:14 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	30	0.0	89970	-0.01	0.03		0.03 - 0.03
2	1595	1.8	88405	-0.19	0.03		0.03 - 0.03
3	1822	2.0	88178	-0.16	0.02		0.02 - 0.02

5	193	0.2	89807	-0.79	1.52		1.52 - 1.52
7	126	0.1	89874	-0.66	2.05		2.04 - 2.05
8	2305	2.6	87695	-0.33	2.45		2.42 - 2.55

Record# 6 Start - End Times 6:15 - 6:29 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	138	0.2	89862	-0.01	0.02		0.02 - 0.02
2	2820	3.1	87180	-0.19	0.02		0.02 - 0.02
3	2207	2.5	87793	-0.16	0.02		0.02 - 0.02
5	120	0.1	89880	-0.79	1.43		1.43 - 1.43
7	69	0.1	89931	-0.66	1.89		1.88 - 1.89
8	2623	2.9	87377	-0.33	1.47		1.44 - 1.53

Record# 7 Start - End Times 6:30 - 6:44 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	78	0.1	89922	-0.01	0.02		0.02 - 0.02
2	3227	3.6	86773	-0.19	0.02		0.02 - 0.02
3	2464	2.7	87536	-0.16	0.02		0.02 - 0.02
5	181	0.2	89819	-0.79	1.36		1.36 - 1.37
7	200	0.2	89800	-0.65	1.90		1.90 - 1.91
8	3330	3.7	86670	-0.36	1.63		1.60 - 1.72

Record# 8 Start - End Times 6:45 - 6:59 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	123	0.1	89877	-0.01	0.02		0.02 - 0.02
2	2030	2.3	87970	-0.19	0.02		0.02 - 0.02
3	1939	2.2	88061	-0.16	0.02		0.02 - 0.02
5	93	0.1	89907	-0.77	1.52		1.52 - 1.52
7	44	0.0	89956	-0.62	2.06		2.06 - 2.06
8	1853	2.1	88147	-0.47	1.74		1.72 - 1.79

Record# 9 Start - End Times 7: 0 - 7:15 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	169	0.2	89831	-0.01	0.01		0.01 - 0.01
2	7111	7.9	82889	-0.19	0.01		0.01 - 0.01
3	1179	1.3	88821	-0.16	0.01		0.01 - 0.01
5	81	0.1	89919	-0.81	0.96		0.96 - 0.96
7	57	0.1	89943	-0.67	1.36		1.36 - 1.36
8	3112	3.5	86888	-0.60	0.90		0.88 - 0.94

Record# 10 Start - End Times 7:16 - 7:30 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	42	0.0	89958	-0.01	0.02		0.02 - 0.02
2	1279	1.4	88721	-0.19	0.02		0.02 - 0.02
3	1775	2.0	88225	-0.16	0.01		0.01 - 0.02
5	95	0.1	89905	-0.78	1.27		1.27 - 1.28
7	86	0.1	89914	-0.64	1.73		1.72 - 1.73
8	2216	2.5	87784	-0.66	1.51		1.49 - 1.57

Record# 11 Start - End Times 7:31 - 7:45 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	392	0.4	89608	-0.01	0.02		0.02 - 0.02
2	1205	1.3	88795	-0.19	0.03		0.03 - 0.03
3	1504	1.7	88496	-0.16	0.02		0.02 - 0.02
5	752	0.8	89248	-0.76	1.82		1.82 - 1.85
7	574	0.6	89426	-0.62	2.58		2.57 - 2.61
8	1886	2.1	88114	-0.64	2.08		2.06 - 2.15

Record# 12 Start - End Times 7:46 - 8: 0 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	225	0.3	89775	-0.01	0.02		0.02 - 0.02
2	553	0.6	89447	-0.19	0.03		0.03 - 0.03
3	511	0.6	89489	-0.16	0.02		0.02 - 0.02
5	415	0.5	89585	-0.76	1.74		1.74 - 1.75
7	231	0.3	89769	-0.65	2.62		2.62 - 2.63
8	1343	1.5	88657	-0.57	1.82		1.80 - 1.86

Record# 13 Start - End Times 8: 1 - 8:15 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	165	0.2	89835	-0.01	0.02		0.02 - 0.02
2	334	0.4	89666	-0.19	0.02		0.02 - 0.02
3	272	0.3	89728	-0.16	0.02		0.02 - 0.02

5	316	0.4	89684	-0.78	1.60		1.60 - 1.61
7	380	0.4	89620	-0.65	2.44		2.44 - 2.46
8	1806	2.0	88194	-0.55	1.64		1.63 - 1.69

Record# 14 Start - End Times 8:16 - 8:31 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	220	0.2	89780	-0.01	0.02		0.02 - 0.02
2	470	0.5	89530	-0.19	0.02		0.02 - 0.02
3	475	0.5	89525	-0.16	0.02		0.02 - 0.02
5	342	0.4	89658	-0.77	1.59		1.59 - 1.60
7	263	0.3	89737	-0.64	2.31		2.31 - 2.32
8	1917	2.1	88083	-0.40	1.63		1.61 - 1.68

Record# 15 Start - End Times 8:32 - 8:46 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	594	0.7	89406	-0.01	0.03		0.03 - 0.03
2	420	0.5	89580	-0.19	0.03		0.03 - 0.03
3	408	0.5	89592	-0.16	0.03		0.03 - 0.03
5	431	0.5	89569	-0.75	1.85		1.85 - 1.87
7	371	0.4	89629	-0.63	3.01		3.00 - 3.03
8	1879	2.1	88121	-0.33	2.09		2.07 - 2.16

Record# 16 Start - End Times 8:47 - 9: 1 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	82	0.1	89918	-0.01	0.02		0.02 - 0.02
2	342	0.4	89658	-0.19	0.03		0.03 - 0.03
3	330	0.4	89670	-0.16	0.02		0.02 - 0.02
5	283	0.3	89717	-0.76	1.57		1.57 - 1.58
7	298	0.3	89702	-0.61	2.49		2.48 - 2.50
8	1770	2.0	88230	-0.29	1.97		1.95 - 2.03

Record# 17 Start - End Times 9: 2 - 9:16 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	241	0.3	89759	-0.01	0.03		0.03 - 0.03
2	153	0.2	89847	-0.19	0.03		0.03 - 0.03
3	229	0.3	89771	-0.16	0.02		0.02 - 0.02
5	504	0.6	89496	-0.75	1.86		1.86 - 1.88
7	442	0.5	89558	-0.63	2.88		2.87 - 2.90
8	2146	2.4	87854	-0.31	2.09		2.06 - 2.16

Record# 18 Start - End Times 9:17 - 9:31 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	99	0.1	89901	0.00	0.03		0.03 - 0.03
2	209	0.2	89791	-0.19	0.03		0.03 - 0.03
3	186	0.2	89814	-0.16	0.02		0.02 - 0.02
5	276	0.3	89724	-0.75	1.59		1.59 - 1.60
7	206	0.2	89794	-0.62	2.66		2.65 - 2.67
8	2204	2.4	87796	-0.46	2.12		2.10 - 2.20

Record# 19 Start - End Times 9:32 - 9:47 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	158	0.2	89842	-0.01	0.02		0.02 - 0.02
2	212	0.2	89788	-0.19	0.03		0.03 - 0.03
3	149	0.2	89851	-0.16	0.02		0.02 - 0.02
5	152	0.2	89848	-0.76	1.37		1.37 - 1.38
7	146	0.2	89854	-0.64	2.05		2.05 - 2.05
8	2266	2.5	87734	-0.46	1.89		1.86 - 1.96

Record# 20 Start - End Times 9:48 - 10: 2 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	112	0.1	89888	-0.01	0.02		0.02 - 0.02
2	304	0.3	89696	-0.19	0.03		0.03 - 0.03
3	292	0.3	89708	-0.16	0.02		0.02 - 0.02
5	525	0.6	89475	-0.76	1.64		1.63 - 1.65
7	462	0.5	89538	-0.63	2.53		2.52 - 2.55
8	2326	2.6	87674	-0.51	1.99		1.97 - 2.07

Record# 21 Start - End Times 10: 3 - 10:12 Count = 57920

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	366	0.6	57554	0.00	0.02		0.02 - 0.03
2	427	0.7	57493	-0.19	0.03		0.03 - 0.03
3	486	0.8	57434	-0.16	0.02		0.02 - 0.02

5	728	1.3	57192	-0.74	2.20	2.19 - 2.24
7	816	1.4	57104	-0.62	3.05	3.03 - 3.11
8	1742	3.0	56178	-0.49	2.13	2.09 - 2.22

This is the summary of the second segment:

Record# 1 Start - End Times 12:28 - 12:42 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	201	0.2	89799	-0.02	0.02		0.02 - 0.02
2	276	0.3	89724	-0.19	0.03		0.03 - 0.03
3	1113	1.2	88887	-0.16	0.02		0.02 - 0.03
5	688	0.8	89312	-0.83	2.10		2.09 - 2.12
7	438	0.5	89562	-0.68	3.32		3.31 - 3.34
8	2435	2.7	87565	-0.18	2.28		2.25 - 2.37

Record# 2 Start - End Times 12:43 - 12:58 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	178	0.2	89822	-0.02	0.02		0.02 - 0.02
2	167	0.2	89833	-0.19	0.03		0.03 - 0.03
3	705	0.8	89295	-0.16	0.02		0.02 - 0.03
5	645	0.7	89355	-0.81	1.86		1.85 - 1.88
7	652	0.7	89348	-0.70	2.97		2.96 - 3.00
8	1702	1.9	88298	-0.62	2.03		2.01 - 2.09

Record# 3 Start - End Times 12:59 - 1:13 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	121	0.1	89879	-0.01	0.02		0.02 - 0.02
2	115	0.1	89885	-0.19	0.03		0.03 - 0.03
3	611	0.7	89389	-0.16	0.02		0.02 - 0.02
5	116	0.1	89884	-0.80	1.45		1.44 - 1.45
7	42	0.0	89958	-0.69	1.91		1.91 - 1.91
8	1342	1.5	88658	-0.47	2.11		2.10 - 2.16

Record# 4 Start - End Times 1:14 - 1:28 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	31	0.0	89969	-0.01	0.02		0.02 - 0.02
2	103	0.1	89897	-0.19	0.02		0.02 - 0.02
3	652	0.7	89348	-0.16	0.01		0.01 - 0.02
5	103	0.1	89897	-0.80	1.25		1.25 - 1.26
7	49	0.1	89951	-0.71	1.55		1.55 - 1.56
8	3279	3.6	86721	-0.44	1.41		1.38 - 1.49

Record# 5 Start - End Times 1:29 - 1:43 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	109	0.1	89891	-0.01	0.02		0.02 - 0.02
2	406	0.5	89594	-0.19	0.02		0.02 - 0.02
3	409	0.5	89591	-0.16	0.02		0.02 - 0.02
5	410	0.5	89590	-0.78	1.47		1.46 - 1.48
7	278	0.3	89722	-0.67	1.98		1.98 - 1.99
8	1937	2.2	88063	-0.39	1.67		1.65 - 1.72

Record# 6 Start - End Times 1:44 - 1:58 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	780	0.9	89220	-0.01	0.01		0.01 - 0.01
2	494	0.5	89506	-0.19	0.02		0.02 - 0.02
3	818	0.9	89182	-0.16	0.02		0.02 - 0.02
5	1356	1.5	88644	-0.78	1.58		1.57 - 1.62
7	1005	1.1	88995	-0.69	1.87		1.86 - 1.90
8	2513	2.8	87487	-0.49	1.36		1.34 - 1.42

Record# 7 Start - End Times 1:59 - 2:14 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	401	0.4	89599	-0.01	0.01		0.01 - 0.01
2	286	0.3	89714	-0.19	0.02		0.02 - 0.02
3	606	0.7	89394	-0.16	0.01		0.01 - 0.01
5	1069	1.2	88931	-0.79	1.33		1.32 - 1.36
7	990	1.1	89010	-0.68	1.56		1.55 - 1.59
8	5203	5.8	84797	-0.48	1.41		1.37 - 1.53

Record# 8 Start - End Times 2:15 - 2:29 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	365	0.4	89635	-0.01	0.02		0.02 - 0.02
2	281	0.3	89719	-0.19	0.02		0.02 - 0.02
3	485	0.5	89515	-0.16	0.02		0.02 - 0.02
5	1179	1.3	88821	-0.77	1.64		1.63 - 1.67
7	1004	1.1	88996	-0.64	2.27		2.25 - 2.30

8	2264	2.5	87736	-0.53	1.87		1.84 -	1.94
Record# 9	Start - End Times		2:30 -	2:40	Count = 63680			
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	509	0.8	63171	-0.01	0.02		0.02 -	0.02
2	425	0.7	63255	-0.19	0.03		0.03 -	0.03
3	784	1.2	62896	-0.16	0.02		0.02 -	0.02
5	643	1.0	63037	-0.74	1.67		1.66 -	1.69
7	538	0.8	63142	-0.61	2.42		2.41 -	2.45
8	1418	2.2	62262	-0.54	2.08		2.06 -	2.15

This is the summary of the third segment:

Record# 1 Start - End Times 10:15 - 10:29 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	129	0.1	89871	-0.02	0.02		0.02 - 0.02
2	367	0.4	89643	-0.19	0.02		0.02 - 0.02
3	922	1.0	89078	-0.16	0.02		0.02 - 0.02
5	422	0.5	89578	-0.88	1.41		1.41 - 1.42
7	295	0.3	89705	-0.74	1.75		1.74 - 1.75
8	4270	4.7	85730	-0.42	1.77		1.72 - 1.89

Record# 2 Start - End Times 10:30 - 10:35 Count = 33520

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	326	1.0	33194	-0.02	0.02		0.02 - 0.02
2	478	1.4	33042	-0.19	0.03		0.03 - 0.03
3	442	1.3	33078	-0.16	0.02		0.02 - 0.02
5	405	1.2	33115	-0.84	1.61		1.60 - 1.64
7	353	1.1	33167	-0.68	2.24		2.23 - 2.28
8	950	2.8	32570	-0.27	2.12		2.09 - 2.20

B.5 D1SN1110

B.5.1 Time Coverage

The first time stamp in this file is 4:53, and the last one is 12:47. The starting time corresponds to MET day 0, hour 11:50, subject I.

B.5.2 Processed Data

Record# 1 Start - End Times 4:53 - 5: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	153	0.2	89847	0.03	0.02		0.02 - 0.02
2	212	0.2	89788	0.11	0.03		0.03 - 0.03
3	120	0.1	89880	-0.08	0.02		0.02 - 0.02
5	264	0.3	89736	-1.09	1.58		1.58 - 1.59
7	996	1.1	89004	-2.43	3.28		3.27 - 3.34
8	188	0.2	89812	-4.04	2.18		2.18 - 2.19

Record# 2 Start - End Times 5: 8 - 5:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	122	0.1	89878	0.03	0.02		0.02 - 0.02
2	103	0.1	89897	0.11	0.02		0.02 - 0.02
3	92	0.1	89908	-0.08	0.02		0.02 - 0.02
5	263	0.3	89737	-0.80	1.52		1.52 - 1.53
7	618	0.7	89382	-2.40	2.86		2.85 - 2.89
8	181	0.2	89819	-3.00	1.88		1.88 - 1.89

Record# 3 Start - End Times 5:23 - 5:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	134	0.1	89866	0.03	0.01		0.01 - 0.01
2	162	0.2	89838	0.12	0.02		0.02 - 0.02
3	19	0.0	89981	-0.08	0.01		0.01 - 0.01
5	43	0.0	89957	-0.21	1.07		1.07 - 1.07
7	147	0.2	89853	-2.39	1.88		1.88 - 1.88
8	85	0.1	89915	-2.51	1.45		1.45 - 1.45

Record# 4 Start - End Times 5:38 - 5:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.01		0.01 - 0.01
2	119	0.1	89881	0.12	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	29	0.0	89971	0.01	1.15		1.15 - 1.15
7	69	0.1	89931	-2.46	2.21		2.21 - 2.21
8	96	0.1	89904	-2.35	1.57		1.57 - 1.57

Record# 5 Start - End Times 5:53 - 6: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.02		0.02 - 0.02
2	134	0.1	89866	0.12	0.02		0.02 - 0.02
3	34	0.0	89966	-0.08	0.02		0.02 - 0.02
5	62	0.1	89938	0.32	1.20		1.20 - 1.21
7	108	0.1	89892	-2.45	2.36		2.36 - 2.37
8	87	0.1	89913	-2.17	1.64		1.63 - 1.64

Record# 6 Start - End Times 6: 8 - 6:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	40	0.0	89960	0.03	0.02		0.02 - 0.02
2	83	0.1	89917	0.12	0.03		0.03 - 0.03
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	28	0.0	89972	0.61	1.45		1.45 - 1.45
7	119	0.1	89881	-2.42	2.69		2.69 - 2.70
8	85	0.1	89915	-1.88	2.09		2.09 - 2.10

Record# 7 Start - End Times 6:23 - 6:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	15	0.0	89985	0.03	0.02		0.02 - 0.02
2	75	0.1	89925	0.12	0.02		0.02 - 0.02
3	25	0.0	89975	-0.08	0.02		0.02 - 0.02
5	25	0.0	89975	0.68	1.22		1.22 - 1.22
7	166	0.2	89834	-2.42	2.29		2.28 - 2.29
8	108	0.1	89892	-1.66	1.87		1.87 - 1.87

Record# 8 Start - End Times 6:38 - 6:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	83	0.1	89917	0.03	0.02		0.02 - 0.02
2	103	0.1	89897	0.12	0.02		0.02 - 0.02
3	52	0.1	89948	-0.08	0.02		0.02 - 0.02
5	186	0.2	89814	-0.10	1.57		1.57 - 1.58
7	447	0.5	89553	-2.42	2.83		2.83 - 2.85
8	251	0.3	89749	-2.11	1.75		1.75 - 1.76

Record# 9 Start - End Times 6:53 - 7: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	1074	1.2	88926	0.02	0.06		0.06 - 0.06
2	533	0.6	89467	0.12	0.06		0.06 - 0.06
3	2182	2.4	87818	-0.09	0.05		0.05 - 0.06
5	1566	1.7	88434	-0.87	2.76		2.73 - 2.83
7	3725	4.1	86275	-2.45	5.14		5.03 - 5.45
8	2351	2.6	87649	-4.15	4.97		4.91 - 5.16

Record# 10 Start - End Times 7: 8 - 7:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	1146	1.3	88854	0.02	0.06		0.06 - 0.06
2	602	0.7	89398	0.12	0.07		0.07 - 0.07
3	2136	2.4	87864	-0.09	0.11		0.11 - 0.11
5	955	1.1	89045	-2.26	2.80		2.79 - 2.85
7	2919	3.2	87081	-2.43	6.18		6.08 - 6.47
8	1086	1.2	88914	-3.20	4.85		4.82 - 4.93

Record# 11 Start - End Times 7:23 - 7:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	849	0.9	89151	0.02	0.04		0.04 - 0.04
2	628	0.7	89372	0.11	0.04		0.04 - 0.04
3	2415	2.7	87585	-0.08	0.06		0.05 - 0.06
5	1570	1.7	88430	-1.65	2.96		2.94 - 3.04
7	4172	4.6	85828	-2.41	5.28		5.16 - 5.64
8	2237	2.5	87763	-3.64	3.79		3.74 - 3.93

Record# 12 Start - End Times 7:38 - 7:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	355	0.4	89645	0.02	0.05		0.05 - 0.05
2	322	0.4	89678	0.11	0.03		0.03 - 0.03
3	4747	5.3	85253	-0.08	0.10		0.10 - 0.11
5	630	0.7	89370	-2.33	2.57		2.56 - 2.60
7	4077	4.5	85923	-2.49	6.06		5.93 - 6.46
8	517	0.6	89483	-4.92	2.60		2.60 - 2.63

Record# 13 Start - End Times 7:53 - 8: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.02		0.02 - 0.02
2	88	0.1	89912	0.11	0.02		0.02 - 0.02
3	19	0.0	89981	-0.08	0.02		0.02 - 0.02
5	23	0.0	89977	-1.63	1.17		1.17 - 1.17
7	74	0.1	89926	-2.51	2.36		2.36 - 2.37
8	86	0.1	89914	-3.80	1.92		1.91 - 1.92

Record# 14 Start - End Times 8: 8 - 8:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	19	0.0	89981	0.03	0.02		0.02 - 0.02
2	83	0.1	89917	0.11	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-1.51	1.18		1.18 - 1.18
7	44	0.0	89956	-2.51	2.30		2.30 - 2.30
8	86	0.1	89914	-3.68	1.95		1.95 - 1.95

Record# 15 Start - End Times 8:23 - 8:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	34	0.0	89966	0.03	0.02		0.02 - 0.02
2	172	0.2	89828	0.11	0.03		0.03 - 0.03
3	157	0.2	89843	-0.08	0.02		0.02 - 0.02
5	146	0.2	89854	-0.78	1.80		1.80 - 1.80
7	594	0.7	89406	-2.47	3.27		3.26 - 3.30
8	199	0.2	89801	-2.97	2.20		2.20 - 2.20

Record# 16 Start - End Times 8:38 - 8:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	31	0.0	89969	0.03	0.02		0.02 - 0.02
2	89	0.1	89911	0.11	0.02		0.02 - 0.02
3	77	0.1	89923	-0.08	0.02		0.02 - 0.02
5	31	0.0	89969	-0.47	1.15		1.15 - 1.15
7	100	0.1	89900	-2.47	2.10		2.10 - 2.10
8	113	0.1	89887	-2.81	1.58		1.58 - 1.59

Record# 17 Start - End Times 8:53 - 9: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	19	0.0	89981	0.03	0.02		0.02 - 0.02
2	78	0.1	89922	0.11	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	29	0.0	89971	-0.40	1.20		1.20 - 1.20
7	148	0.2	89852	-2.45	2.19		2.19 - 2.20
8	71	0.1	89929	-2.72	1.74		1.73 - 1.74

Record# 18 Start - End Times 9: 8 - 9:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	52	0.1	89948	0.03	0.01		0.01 - 0.01
2	87	0.1	89913	0.11	0.02		0.02 - 0.02
3	27	0.0	89973	-0.08	0.02		0.02 - 0.02
5	97	0.1	89903	-0.52	1.31		1.31 - 1.31
7	325	0.4	89675	-2.46	2.51		2.50 - 2.52
8	130	0.1	89870	-2.91	1.92		1.91 - 1.92

Record# 19 Start - End Times 9:23 - 9:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.02		0.02 - 0.02
2	75	0.1	89925	0.11	0.03		0.03 - 0.03
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	37	0.0	89963	-1.09	1.32		1.32 - 1.32
7	97	0.1	89903	-2.47	2.69		2.69 - 2.70
8	75	0.1	89925	-3.33	2.52		2.52 - 2.52

Record# 20 Start - End Times 9:38 - 9:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	119	0.1	89881	0.03	0.02		0.02 - 0.02
2	168	0.2	89832	0.11	0.03		0.03 - 0.03
3	97	0.1	89903	-0.08	0.02		0.02 - 0.02
5	490	0.5	89510	-0.83	1.43		1.43 - 1.44
7	614	0.7	89386	-2.46	2.51		2.50 - 2.54
8	299	0.3	89701	-2.93	2.22		2.21 - 2.23

Record# 21 Start - End Times 9:53 - 10: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.01		0.01 - 0.01
2	75	0.1	89925	0.11	0.01		0.01 - 0.01
3	32	0.0	89968	-0.08	0.01		0.01 - 0.01
5	51	0.1	89949	-1.98	0.76		0.76 - 0.76
7	35	0.0	89965	-2.51	1.59		1.59 - 1.59
8	87	0.1	89913	-3.20	1.07		1.07 - 1.07

Record# 22 Start - End Times 10: 8 - 10:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	104	0.1	89896	0.03	0.01		0.01 - 0.01
2	75	0.1	89925	0.11	0.01		0.01 - 0.01
3	15	0.0	89985	-0.08	0.01		0.01 - 0.01
5	15	0.0	89985	-2.28	0.51		0.51 - 0.51
7	15	0.0	89985	-2.52	1.10		1.10 - 1.10
8	75	0.1	89925	-3.53	0.53		0.53 - 0.53

Record# 23 Start - End Times 10:23 - 10:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	62	0.1	89938	0.03	0.01		0.01 - 0.01
2	109	0.1	89891	0.11	0.02		0.02 - 0.02
3	66	0.1	89934	-0.08	0.02		0.02 - 0.02
5	85	0.1	89915	-2.75	1.15		1.15 - 1.16
7	172	0.2	89828	-2.51	2.30		2.30 - 2.31
8	146	0.2	89854	-3.98	1.55		1.55 - 1.55

Record# 24 Start - End Times 10:38 - 10:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	71	0.1	89929	0.03	0.02		0.02 - 0.02
2	100	0.1	89900	0.11	0.02		0.02 - 0.02
3	58	0.1	89942	-0.08	0.02		0.02 - 0.02
5	201	0.2	89799	-2.07	1.39		1.39 - 1.40
7	435	0.5	89565	-2.45	2.56		2.56 - 2.58
8	154	0.2	89846	-4.02	1.94		1.94 - 1.95

Record# 25 Start - End Times 10:53 - 11: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	67	0.1	89933	0.03	0.01		0.01 - 0.01
2	75	0.1	89925	0.11	0.02		0.02 - 0.02
3	19	0.0	89981	-0.08	0.02		0.02 - 0.02
5	70	0.1	89930	-1.71	1.05		1.05 - 1.05
7	390	0.4	89610	-2.44	1.97		1.97 - 1.99
8	75	0.1	89925	-3.70	1.63		1.63 - 1.64

Record# 26 Start - End Times 11: 8 - 11:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	31	0.0	89969	0.03	0.02		0.02 - 0.02
2	91	0.1	89909	0.11	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	67	0.1	89933	-1.27	1.42		1.41 - 1.42
7	223	0.2	89777	-2.43	2.66		2.66 - 2.67
8	97	0.1	89903	-3.36	1.81		1.81 - 1.81

Record# 27 Start - End Times 11:23 - 11:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.01		0.01 - 0.01
2	75	0.1	89925	0.11	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-1.44	1.03		1.03 - 1.03
7	43	0.0	89957	-2.45	1.89		1.89 - 1.89
8	75	0.1	89925	-3.55	1.70		1.70 - 1.70

Record# 28 Start - End Times 11:38 - 11:52 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.01		0.01 - 0.01
2	75	0.1	89925	0.11	0.02		0.02 - 0.02
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-1.19	1.03		1.03 - 1.03
7	15	0.0	89985	-2.43	1.92		1.92 - 1.92
8	71	0.1	89929	-3.33	1.56		1.56 - 1.57

Record# 29 Start - End Times 11:53 - 12: 7 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	25	0.0	89975	0.03	0.02		0.02 - 0.02
2	102	0.1	89898	0.11	0.03		0.03 - 0.03
3	15	0.0	89985	-0.08	0.02		0.02 - 0.02
5	38	0.0	89962	-0.92	1.49		1.49 - 1.49
7	65	0.1	89935	-2.39	2.71		2.71 - 2.71
8	134	0.1	89866	-3.11	2.82		2.82 - 2.82

Record# 30 Start - End Times 12: 8 - 12:22 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	0.03	0.02		0.02 - 0.02
2	75	0.1	89925	0.11	0.03		0.03 - 0.03
3	19	0.0	89981	-0.08	0.02		0.02 - 0.02
5	15	0.0	89985	-1.41	1.27		1.27 - 1.27
7	47	0.1	89953	-2.42	2.10		2.10 - 2.10
8	101	0.1	89899	-3.60	2.80		2.80 - 2.80

Record# 31 Start - End Times 12:23 - 12:37 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	15	0.0	89985	0.03	0.01		0.01 -	0.01
2	75	0.1	89925	0.11	0.02		0.02 -	0.02
3	15	0.0	89985	-0.08	0.02		0.02 -	0.02
5	20	0.0	89980	-1.75	1.16		1.16 -	1.16
7	48	0.1	89952	-2.45	2.23		2.23 -	2.23
8	71	0.1	89929	-3.86	2.16		2.16 -	2.16

Record# 32 Start - End Times 12:38 - 12:47 Count = 49040

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	592	1.2	48448	0.03	0.02		0.02 -	0.02
2	503	1.0	48537	0.11	0.02		0.02 -	0.02
3	451	0.9	48589	-0.08	0.02		0.02 -	0.02
5	864	1.8	48176	-1.97	2.63		2.61 -	2.70
7	942	1.9	48098	-2.46	2.63		2.61 -	2.71
8	785	1.6	48255	-4.01	2.99		2.96 -	3.06

B.6 D1SN1114

B.6.1 Time Coverage

The first time stamp in this file is 1:47 in rec.11, and the last time stamp is 9:40 in rec.8815. The starting time corresponds to MET day 4, hour 8:45, subject I.

B.6.2 Processed Data

Below are statistics for the cleaned file.

Record# 1 Start - End Times 1:47 - 2: 1 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	434	0.5	89566	0.03	0.02		0.02 - 0.02
2	205	0.2	89795	0.10	0.04		0.04 - 0.04
3	138	0.2	89862	-0.08	0.03		0.03 - 0.03
5	401	0.4	89599	-3.21	2.79		2.78 - 2.81
7	1282	1.4	88718	-2.47	3.96		3.94 - 4.05
8	529	0.6	89471	-5.77	3.51		3.50 - 3.55

Record# 2 Start - End Times 2: 2 - 2:16 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	234	0.3	89766	0.03	0.03		0.03 - 0.03
2	204	0.2	89796	0.10	0.04		0.04 - 0.04
3	122	0.1	89878	-0.08	0.03		0.03 - 0.03
5	525	0.6	89475	-2.43	2.34		2.34 - 2.36
7	1538	1.7	88462	-2.44	4.17		4.14 - 4.28
8	328	0.4	89672	-4.97	3.61		3.60 - 3.63

Record# 3 Start - End Times 2:17 - 2:31 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	25	0.0	89975	0.03	0.02		0.02 - 0.02
2	111	0.1	89889	0.10	0.03		0.03 - 0.03
3	74	0.1	89926	-0.08	0.02		0.02 - 0.02
5	20	0.0	89980	-3.83	1.74		1.74 - 1.74
7	153	0.2	89847	-2.47	3.23		3.23 - 3.24
8	393	0.4	89607	-6.07	2.74		2.73 - 2.75

Record# 4 Start - End Times 2:32 - 2:46 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	98	0.1	89902	0.03	0.02		0.02 - 0.02
2	182	0.2	89818	0.10	0.04		0.04 - 0.04
3	59	0.1	89941	-0.09	0.03		0.03 - 0.03
5	252	0.3	89748	-4.10	2.21		2.21 - 2.22
7	1119	1.2	88881	-2.41	3.78		3.76 - 3.85
8	291	0.3	89709	-6.05	3.47		3.46 - 3.48

Record# 5 Start - End Times 2:47 - 3: 1 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	22	0.0	89978	0.03	0.02		0.02 - 0.02
2	86	0.1	89914	0.09	0.03		0.03 - 0.03
3	21	0.0	89979	-0.09	0.02		0.02 - 0.02
5	50	0.1	89950	-4.60	1.86		1.86 - 1.86
7	319	0.4	89681	-2.45	3.65		3.64 - 3.67
8	230	0.3	89770	-6.60	2.95		2.95 - 2.96

Record# 6 Start - End Times 3: 2 - 3:16 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	153	0.2	89847	0.03	0.02		0.02 - 0.02
2	116	0.1	89884	0.09	0.03		0.03 - 0.03
3	117	0.1	89883	-0.09	0.02		0.02 - 0.02
5	243	0.3	89757	-3.10	2.10		2.10 - 2.11
7	264	0.3	89736	-2.41	2.57		2.57 - 2.58
8	543	0.6	89457	-6.57	3.00		2.99 - 3.02

Record# 7 Start - End Times 3:17 - 3:31 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	118	0.1	89882	0.03	0.02		0.02 - 0.02
2	99	0.1	89901	0.09	0.04		0.04 - 0.04
3	91	0.1	89909	-0.09	0.02		0.02 - 0.02
5	37	0.0	89963	-2.73	1.72		1.72 - 1.72
7	41	0.0	89959	-2.43	2.58		2.58 - 2.58
8	298	0.3	89702	-7.16	3.09		3.09 - 3.11

Record# 8 Start - End Times 3:32 - 3:46 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	45	0.1	89955	0.03	0.02		0.02 - 0.02
2	123	0.1	89877	0.09	0.03		0.03 - 0.03
3	96	0.1	89904	-0.09	0.03		0.03 - 0.03
5	116	0.1	89884	-2.24	2.03		2.03 - 2.03
7	403	0.4	89597	-2.43	3.12		3.11 - 3.14
8	318	0.4	89682	-6.40	2.99		2.99 - 3.01

Record# 9 Start - End Times 3:47 - 4: 1 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	162	0.2	89838	0.03	0.02		0.02 - 0.02
2	71	0.1	89929	0.09	0.03		0.03 - 0.03
3	24	0.0	89976	-0.09	0.02		0.02 - 0.02
5	86	0.1	89914	-0.46	1.42		1.42 - 1.42
7	213	0.2	89787	-2.48	2.26		2.26 - 2.27
8	174	0.2	89826	-5.44	2.01		2.01 - 2.02

Record# 10 Start - End Times 4: 2 - 4:17 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	32	0.0	89968	0.02	0.03		0.03 - 0.03
2	99	0.1	89901	0.09	0.04		0.04 - 0.04
3	40	0.0	89960	-0.09	0.02		0.02 - 0.02
5	16	0.0	89984	-1.51	1.62		1.62 - 1.62
7	94	0.1	89906	-2.43	2.61		2.61 - 2.61
8	164	0.2	89836	-5.97	3.44		3.43 - 3.45

Record# 11 Start - End Times 4:18 - 4:32 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	49	0.1	89951	0.02	0.03		0.03 - 0.03
2	109	0.1	89891	0.09	0.03		0.03 - 0.03
3	47	0.1	89953	-0.09	0.03		0.03 - 0.03
5	116	0.1	89884	-1.16	1.64		1.64 - 1.64
7	257	0.3	89743	-2.43	2.96		2.96 - 2.98
8	115	0.1	89885	-5.36	3.02		3.01 - 3.02

Record# 12 Start - End Times 4:33 - 4:47 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	382	0.4	89618	0.02	0.03		0.03 - 0.03
2	264	0.3	89736	0.09	0.04		0.04 - 0.04
3	282	0.3	89718	-0.09	0.03		0.03 - 0.03
5	285	0.3	89715	-0.55	2.05		2.05 - 2.06
7	284	0.3	89716	-2.39	3.36		3.35 - 3.38
8	496	0.6	89504	-4.66	3.90		3.89 - 3.94

Record# 13 Start - End Times 4:48 - 5: 2 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	700	0.8	89300	0.02	0.03		0.03 - 0.03
2	450	0.5	89550	0.09	0.05		0.05 - 0.05
3	489	0.5	89511	-0.09	0.03		0.03 - 0.03
5	246	0.3	89754	-1.63	2.17		2.17 - 2.18
7	319	0.4	89681	-2.40	3.53		3.52 - 3.55
8	744	0.8	89256	-5.80	4.20		4.18 - 4.25

Record# 14 Start - End Times 5: 3 - 5:17 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2624	2.9	87376	0.02	0.03		0.03 - 0.03
2	467	0.5	89533	0.09	0.04		0.04 - 0.04
3	720	0.8	89280	-0.09	0.03		0.03 - 0.03
5	611	0.7	89389	-0.88	2.70		2.69 - 2.72
7	750	0.8	89250	-2.42	3.31		3.30 - 3.35
8	813	0.9	89187	-6.27	3.98		3.96 - 4.03

Record# 15 Start - End Times 5:18 - 5:32 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2168	2.4	87832	0.02	0.03		0.03 - 0.03
2	184	0.2	89816	0.09	0.04		0.04 - 0.04
3	585	0.7	89415	-0.09	0.03		0.03 - 0.03
5	174	0.2	89826	1.06	1.86		1.86 - 1.87
7	350	0.4	89650	-2.42	3.41		3.40 - 3.43
8	766	0.9	89234	-6.00	3.01		3.00 - 3.05

Record# 16 Start - End Times 5:33 - 5:47 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2503	2.8	87497	0.03	0.02		0.02 - 0.02
2	355	0.4	89645	0.09	0.03		0.03 - 0.03
3	585	0.7	89415	-0.09	0.02		0.02 - 0.02
5	387	0.4	89613	1.19	1.63		1.63 - 1.64
7	514	0.6	89486	-2.43	2.51		2.50 - 2.53
8	1099	1.2	88901	-5.76	2.64		2.62 - 2.69

Record# 17 Start - End Times 5:48 - 6: 2 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	23348	25.9	66652	0.03	0.02		0.02 - 0.03
2	1057	1.2	88943	0.09	0.02		0.02 - 0.02
3	839	0.9	89161	-0.09	0.02		0.02 - 0.02
5	906	1.0	89094	-2.15	1.49		1.48 - 1.51
7	639	0.7	89361	-2.46	2.28		2.27 - 2.31
8	881	1.0	89119	-3.74	2.19		2.18 - 2.22

Record# 18 Start - End Times 6: 3 - 6:17 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	24905	27.7	65095	0.02	0.02		0.02 - 0.02
2	465	0.5	89535	0.09	0.02		0.02 - 0.02
3	968	1.1	89032	-0.09	0.02		0.02 - 0.02
5	348	0.4	89652	-2.37	1.21		1.20 - 1.21
7	481	0.5	89519	-2.47	2.20		2.19 - 2.21
8	922	1.0	89078	-3.92	1.97		1.96 - 2.00

Record# 19 Start - End Times 6:19 - 6:33 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	4389	4.9	85611	0.02	0.02		0.02 - 0.02
2	482	0.5	89518	0.09	0.03		0.03 - 0.03
3	632	0.7	89368	-0.09	0.02		0.02 - 0.02
5	418	0.5	89582	-1.60	1.40		1.39 - 1.41
7	174	0.2	89826	-2.40	2.16		2.16 - 2.16
8	739	0.8	89261	-3.89	2.80		2.78 - 2.83

Record# 20 Start - End Times 6:34 - 6:48 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2921	3.2	87079	0.02	0.03		0.03 - 0.03
2	492	0.5	89508	0.09	0.04		0.04 - 0.04
3	495	0.6	89505	-0.09	0.03		0.03 - 0.03
5	290	0.3	89710	-0.75	1.63		1.63 - 1.64
7	310	0.3	89690	-2.36	3.09		3.09 - 3.11
8	632	0.7	89368	-3.10	3.14		3.13 - 3.18

Record# 21 Start - End Times 6:49 - 7: 3 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	1044	1.2	88956	0.02	0.03		0.03 - 0.03
2	505	0.6	89495	0.09	0.04		0.04 - 0.04
3	553	0.6	89447	-0.09	0.03		0.03 - 0.03
5	449	0.5	89551	-0.57	2.08		2.08 - 2.10
7	284	0.3	89716	-2.36	3.41		3.40 - 3.43
8	744	0.8	89256	-3.04	3.64		3.63 - 3.69

Record# 22 Start - End Times 7: 4 - 7:18 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	1893	2.1	88107	0.03	0.02		0.02 - 0.02
2	609	0.7	89391	0.09	0.03		0.03 - 0.03
3	576	0.6	89424	-0.09	0.02		0.02 - 0.02
5	405	0.4	89595	-0.99	1.56		1.56 - 1.57
7	202	0.2	89798	-2.39	2.64		2.64 - 2.65
8	852	0.9	89148	-3.54	2.97		2.95 - 3.01

Record# 23 Start - End Times 7:19 - 7:33 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	3720	4.1	86280	0.03	0.01		0.01 - 0.02
2	602	0.7	89398	0.09	0.02		0.02 - 0.02
3	584	0.6	89416	-0.09	0.02		0.02 - 0.02
5	405	0.4	89595	-1.39	1.00		1.00 - 1.01
7	309	0.3	89691	-2.45	1.96		1.96 - 1.97
8	808	0.9	89192	-3.97	2.02		2.02 - 2.05

Record# 24 Start - End Times 7:34 - 7:48 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	4826	5.4	85174	0.03	0.02		0.02 - 0.02
2	366	0.4	89634	0.09	0.03		0.03 - 0.03
3	632	0.7	89368	-0.09	0.02		0.02 - 0.02
5	283	0.3	89717	-1.27	1.44		1.44 - 1.45
7	190	0.2	89810	-2.41	2.56		2.55 - 2.56
8	782	0.9	89218	-3.67	2.89		2.88 - 2.93

Record# 25 Start - End Times 7:49 - 8: 4 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	7233	8.0	82767	0.03	0.02		0.02 - 0.02
2	535	0.6	89465	0.09	0.03		0.03 - 0.03
3	726	0.8	89274	-0.09	0.02		0.02 - 0.02
5	406	0.5	89594	-0.92	1.27		1.26 - 1.27
7	184	0.2	89816	-2.40	2.00		2.00 - 2.00
8	979	1.1	89021	-3.41	2.44		2.43 - 2.48

Record# 26 Start - End Times 8: 5 - 8:19 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	10920	12.1	79080	0.03	0.02		0.02 - 0.02
2	575	0.6	89425	0.09	0.03		0.03 - 0.03
3	1057	1.2	88943	-0.09	0.02		0.02 - 0.02
5	330	0.4	89670	-0.59	1.23		1.23 - 1.24
7	482	0.5	89518	-2.39	1.88		1.88 - 1.90
8	670	0.7	89330	-3.04	2.24		2.24 - 2.27

Record# 27 Start - End Times 8:20 - 8:34 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	3825	4.3	86175	0.03	0.02		0.02 - 0.02
2	859	1.0	89141	0.09	0.03		0.03 - 0.03
3	812	0.9	89188	-0.09	0.02		0.02 - 0.02
5	605	0.7	89395	-0.45	1.36		1.35 - 1.37
7	310	0.3	89690	-2.40	2.19		2.18 - 2.20
8	1024	1.1	88976	-2.85	2.44		2.42 - 2.48

Record# 28 Start - End Times 8:35 - 8:49 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	3313	3.7	86687	0.02	0.02		0.02 - 0.02
2	606	0.7	89394	0.09	0.03		0.03 - 0.03
3	893	1.0	89107	-0.09	0.02		0.02 - 0.02
5	537	0.6	89463	-0.17	1.66		1.65 - 1.67
7	436	0.5	89564	-2.36	2.82		2.82 - 2.84
8	1065	1.2	88935	-2.74	2.97		2.95 - 3.02

Record# 29 Start - End Times 8:50 - 9: 4 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	3113	3.5	86887	0.03	0.01		0.01 - 0.02
2	897	1.0	89103	0.09	0.02		0.02 - 0.02
3	788	0.9	89212	-0.09	0.02		0.02 - 0.02
5	596	0.7	89404	-0.55	1.08		1.08 - 1.09
7	433	0.5	89567	-2.42	2.05		2.04 - 2.06
8	865	1.0	89135	-3.17	1.87		1.86 - 1.89

Record# 30 Start - End Times 9: 5 - 9:19 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2408	2.7	87592	0.02	0.02		0.02 - 0.02
2	492	0.5	89508	0.09	0.03		0.03 - 0.03
3	784	0.9	89216	-0.09	0.02		0.02 - 0.02
5	379	0.4	89621	-1.19	1.30		1.29 - 1.30
7	315	0.4	89685	-2.44	2.45		2.45 - 2.46
8	758	0.8	89242	-3.83	2.25		2.24 - 2.28

Record# 31 Start - End Times 9:20 - 9:35 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	7600	8.4	82400	0.02	0.02		0.02 - 0.03
2	412	0.5	89588	0.09	0.03		0.03 - 0.03
3	941	1.0	89059	-0.09	0.02		0.02 - 0.02
5	579	0.6	89421	-0.97	1.49		1.49 - 1.51
7	328	0.4	89672	-2.41	2.79		2.78 - 2.80
8	1126	1.3	88874	-3.43	2.90		2.89 - 2.96

Record# 32 Start - End Times 9:36 - 9:39 Count = 29200

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	422	1.4	28778	0.02	0.03		0.03 - 0.03
2	206	0.7	28994	0.09	0.05		0.05 - 0.05
3	338	1.2	28862	-0.09	0.03		0.03 - 0.03
5	190	0.7	29010	-0.63	2.19		2.18 - 2.21
7	147	0.5	29053	-2.38	3.90		3.89 - 3.92
8	392	1.3	28808	-3.17	4.25		4.22 - 4.34

B.7 D1SN1125

B.7.1 Time Coverage

The first time stamp in this file is 9:19, and the last one 4:55. The starting time corresponds to MET day 0, hour 4:17, subject J.

B.7.2 Processed Data

Record# 1 Start - End Times 9:19 - 9:34 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	2559	2.8	87441	-0.01	0.02		0.02 - 0.02
2	4468	5.0	85532	-0.19	0.02		0.02 - 0.03
3	2882	3.2	87118	-0.16	0.02		0.02 - 0.02
5	2734	3.0	87266	-0.77	1.87		1.84 - 1.95
7	2527	2.8	87473	-0.62	2.47		2.44 - 2.58
8	5401	6.0	84599	-0.58	1.74		1.69 - 1.89

Record# 2 Start - End Times 9:35 - 9:49 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	124	0.1	89876	-0.01	0.02		0.02 - 0.02
2	4282	4.8	85718	-0.19	0.02		0.02 - 0.02
3	674	0.7	89326	-0.16	0.02		0.02 - 0.02
5	299	0.3	89701	-0.78	1.53		1.53 - 1.54
7	84	0.1	89916	-0.65	1.99		1.99 - 2.00
8	2227	2.5	87773	-0.74	1.54		1.52 - 1.60

Record# 3 Start - End Times 9: 6 - 10: 4 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	402	0.4	89598	-0.02	0.02		0.02 - 0.02
2	1868	2.1	88132	-0.19	0.03		0.03 - 0.03
3	1052	1.2	88948	-0.16	0.02		0.02 - 0.02
5	379	0.4	89621	-0.79	1.63		1.63 - 1.64
7	333	0.4	89667	-0.66	2.05		2.04 - 2.06
8	2494	2.8	87506	-0.71	1.79		1.76 - 1.86

Record# 4 Start - End Times 10: 5 - 10:19 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	321	0.4	89679	-0.02	0.02		0.02 - 0.02
2	1841	2.0	88159	-0.19	0.02		0.02 - 0.02
3	820	0.9	89180	-0.16	0.02		0.02 - 0.02
5	447	0.5	89553	-0.80	1.58		1.57 - 1.59
7	222	0.2	89778	-0.67	2.15		2.15 - 2.16
8	1914	2.1	88086	-0.69	1.67		1.65 - 1.72

Record# 5 Start - End Times 10:20 - 10:34 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	88	0.1	89912	-0.02	0.02		0.02 - 0.02
2	1848	2.1	88152	-0.19	0.03		0.03 - 0.03
3	814	0.9	89186	-0.16	0.02		0.02 - 0.02
5	265	0.3	89735	-0.80	1.77		1.76 - 1.77
7	140	0.2	89860	-0.66	2.45		2.45 - 2.45
8	1916	2.1	88084	-0.64	2.16		2.13 - 2.23

Record# 6 Start - End Times 10:35 - 10:50 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	247	0.3	89753	-0.02	0.02		0.02 - 0.02
2	3093	3.4	86907	-0.19	0.03		0.03 - 0.03
3	853	0.9	89147	-0.16	0.02		0.02 - 0.02
5	189	0.2	89811	-0.81	1.70		1.70 - 1.71
7	69	0.1	89931	-0.68	2.37		2.37 - 2.37
8	2265	2.5	87735	-0.58	2.05		2.02 - 2.12

Record# 7 Start - End Times 10:51 - 11: 5 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	77	0.1	89923	-0.02	0.02		0.02 - 0.02
2	1998	2.2	88002	-0.19	0.02		0.02 - 0.02
3	773	0.9	89227	-0.16	0.02		0.02 - 0.02
5	180	0.2	89820	-0.82	1.55		1.55 - 1.56
7	67	0.1	89933	-0.69	2.12		2.12 - 2.13
8	2163	2.4	87837	-0.58	1.60		1.58 - 1.66

Record# 8 Start - End Times 11: 6 - 11:20 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	115	0.1	89885	-0.02	0.02		0.02 - 0.02
2	1966	2.2	88034	-0.19	0.03		0.03 - 0.03
3	737	0.8	89263	-0.16	0.02		0.02 - 0.02
5	213	0.2	89787	-0.81	1.75		1.75 - 1.76
7	131	0.1	89869	-0.68	2.41		2.41 - 2.41
8	1897	2.1	88103	-0.59	1.77		1.75 - 1.83

Record# 9 Start - End Times 11:21 - 11:35 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	339	0.4	89661	-0.02	0.03		0.03 - 0.03
2	2270	2.5	87730	-0.19	0.03		0.03 - 0.03
3	604	0.7	89396	-0.16	0.03		0.03 - 0.03
5	617	0.7	89383	-0.80	2.04		2.03 - 2.06
7	516	0.6	89484	-0.66	3.10		3.09 - 3.13
8	1776	2.0	88224	-0.58	1.94		1.92 - 1.99

Record# 10 Start - End Times 11:36 - 12: 0 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	450	0.5	89550	-0.02	0.02		0.02 - 0.02
2	2362	2.6	87638	-0.19	0.03		0.03 - 0.03
3	806	0.9	89194	-0.16	0.03		0.02 - 0.03
5	691	0.8	89309	-0.81	1.91		1.90 - 1.93
7	492	0.5	89508	-0.68	2.83		2.82 - 2.85
8	2455	2.7	87545	-0.58	2.07		2.04 - 2.15

Record# 11 Start - End Times 11:51 - 12: 6 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	314	0.3	89686	-0.02	0.03		0.03 - 0.03
2	2225	2.5	87775	-0.19	0.03		0.03 - 0.03
3	502	0.6	89498	-0.16	0.02		0.02 - 0.02
5	565	0.6	89435	-0.81	1.92		1.92 - 1.94
7	398	0.4	89602	-0.66	2.78		2.77 - 2.80
8	1960	2.2	88040	-0.57	2.11		2.08 - 2.17

Record# 12 Start - End Times 12: 7 - 12:21 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	199	0.2	89801	-0.02	0.02		0.02 - 0.02
2	1126	1.3	88874	-0.19	0.03		0.03 - 0.03
3	478	0.5	89522	-0.16	0.02		0.02 - 0.02
5	150	0.2	89850	-0.82	1.62		1.62 - 1.63
7	131	0.1	89869	-0.62	2.21		2.21 - 2.22
8	1713	1.9	88287	-0.59	2.34		2.32 - 2.41

Record# 13 Start - End Times 12:22 - 12:36 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	197	0.2	89803	-0.02	0.02		0.02 - 0.02
2	1264	1.4	88736	-0.19	0.03		0.03 - 0.03
3	326	0.4	89674	-0.16	0.02		0.02 - 0.02
5	236	0.3	89764	-0.81	1.61		1.60 - 1.61
7	175	0.2	89825	-0.55	2.12		2.12 - 2.12
8	1747	1.9	88253	-0.58	2.30		2.28 - 2.37

Record# 14 Start - End Times 12:37 - 12:51 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	121	0.1	89879	-0.02	0.02		0.02 - 0.02
2	638	0.7	89362	-0.19	0.03		0.03 - 0.03
3	294	0.3	89706	-0.16	0.02		0.02 - 0.02
5	341	0.4	89659	-0.82	1.71		1.71 - 1.72
7	328	0.4	89672	-0.67	2.45		2.45 - 2.47
8	1788	2.0	88212	-0.57	2.07		2.05 - 2.13

Record# 15 Start - End Times 12:52 - 1: 6 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	2470	2.7	87530	-0.02	0.03		0.03 -	0.03
2	1132	1.3	88868	-0.19	0.03		0.03 -	0.03
3	624	0.7	89376	-0.16	0.03		0.03 -	0.03
5	1572	1.7	88428	-0.82	2.06		2.04 -	2.11
7	1592	1.8	88408	-0.69	3.24		3.21 -	3.32
8	5902	6.6	84098	-0.56	2.33		2.25 -	2.55
Record# 16	Start - End Times	1: 7 -	1:22	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	2207	2.5	87793	-0.02	0.00		0.00 -	0.00
2	80	0.1	89920	-0.19	0.00		0.00 -	0.00
3	16	0.0	89984	-0.16	0.00		0.00 -	0.00
5	16	0.0	89984	-0.89	0.06		0.06 -	0.06
7	232	0.3	89768	-0.88	0.08		0.08 -	0.08
8	1084	1.2	88916	-0.73	0.15		0.14 -	0.15
Record# 17	Start - End Times	1:23 -	1:37	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	15	0.0	89985	-0.02	0.00		0.00 -	0.00
2	75	0.1	89925	-0.19	0.00		0.00 -	0.00
3	15	0.0	89985	-0.16	0.00		0.00 -	0.00
5	15	0.0	89985	-0.89	0.06		0.06 -	0.06
7	15	0.0	89985	-0.89	0.07		0.07 -	0.07
8	686	0.8	89314	-0.72	0.14		0.14 -	0.15
Record# 18	Start - End Times	1:38 -	1:52	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	15	0.0	89985	-0.02	0.00		0.00 -	0.00
2	75	0.1	89925	-0.20	0.00		0.00 -	0.00
3	15	0.0	89985	-0.16	0.00		0.00 -	0.00
5	15	0.0	89985	-0.89	0.06		0.06 -	0.06
7	15	0.0	89985	-0.91	0.09		0.09 -	0.09
8	871	1.0	89129	-0.74	0.15		0.15 -	0.15
Record# 19	Start - End Times	1:53 -	2: 7	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	19	0.0	89981	-0.02	0.00		0.00 -	0.00
2	174	0.2	89826	-0.20	0.00		0.00 -	0.00
3	19	0.0	89981	-0.16	0.00		0.00 -	0.00
5	20	0.0	89980	-0.90	0.07		0.07 -	0.07
7	31	0.0	89969	-0.91	0.10		0.10 -	0.10
8	746	0.8	89254	-0.74	0.15		0.15 -	0.15
Record# 20	Start - End Times	2: 8 -	2:22	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	23	0.0	89977	-0.02	0.00		0.00 -	0.00
2	68	0.1	89932	-0.20	0.00		0.00 -	0.00
3	23	0.0	89977	-0.16	0.00		0.00 -	0.00
5	20	0.0	89980	-0.90	0.07		0.07 -	0.07
7	20	0.0	89980	-0.91	0.10		0.10 -	0.10
8	747	0.8	89253	-0.72	0.14		0.14 -	0.15
Record# 21	Start - End Times	2:23 -	2:38	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	16	0.0	89984	-0.02	0.00		0.00 -	0.00
2	3163	3.5	86837	-0.20	0.00		0.00 -	0.00
3	16	0.0	89984	-0.16	0.00		0.00 -	0.00
5	16	0.0	89984	-0.90	0.06		0.06 -	0.06
7	331	0.4	89669	-0.91	0.10		0.10 -	0.11
8	751	0.8	89249	-0.71	0.14		0.14 -	0.14
Record# 22	Start - End Times	2:39 -	2:53	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER
1	15	0.0	89985	-0.02	0.00		0.00 -	0.00
2	75	0.1	89925	-0.20	0.00		0.00 -	0.00
3	15	0.0	89985	-0.16	0.00		0.00 -	0.00
5	15	0.0	89985	-0.90	0.08		0.08 -	0.08
7	15	0.0	89985	-0.91	0.09		0.09 -	0.09
8	891	1.0	89109	-0.68	0.14		0.14 -	0.14
Record# 23	Start - End Times	2:54 -	3: 8	Count = 90000				
CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER -	UPPER

1	19	0.0	89981	-0.02	0.00		0.00 - 0.00
2	79	0.1	89921	-0.20	0.00		0.00 - 0.00
3	19	0.0	89981	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.90	0.08		0.08 - 0.08
7	19	0.0	89981	-0.91	0.09		0.09 - 0.09
8	1222	1.4	88778	-0.65	0.16		0.16 - 0.16

Record# 24 Start - End Times 3: 9 - 3:23 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	-0.02	0.00		0.00 - 0.00
2	75	0.1	89925	-0.20	0.00		0.00 - 0.00
3	15	0.0	89985	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.89	0.06		0.06 - 0.06
7	15	0.0	89985	-0.91	0.09		0.09 - 0.09
8	2193	2.4	87807	-0.64	0.12		0.12 - 0.12

Record# 25 Start - End Times 3:24 - 3:38 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	19	0.0	89981	-0.02	0.00		0.00 - 0.00
2	75	0.1	89925	-0.20	0.00		0.00 - 0.00
3	19	0.0	89981	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.89	0.08		0.08 - 0.08
7	19	0.0	89981	-0.91	0.10		0.10 - 0.10
8	1554	1.7	88446	-0.64	0.12		0.12 - 0.12

Record# 26 Start - End Times 3:39 - 3:53 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	15	0.0	89985	-0.02	0.00		0.00 - 0.00
2	79	0.1	89921	-0.20	0.00		0.00 - 0.00
3	15	0.0	89985	-0.16	0.00		0.00 - 0.00
5	15	0.0	89985	-0.89	0.07		0.07 - 0.07
7	15	0.0	89985	-0.91	0.10		0.10 - 0.10
8	2254	2.5	87746	-0.63	0.11		0.11 - 0.12

Record# 27 Start - End Times 3:54 - 4: 9 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	16	0.0	89984	-0.02	0.00		0.00 - 0.00
2	4469	5.0	85531	-0.20	0.00		0.00 - 0.00
3	16	0.0	89984	-0.16	0.00		0.00 - 0.00
5	16	0.0	89984	-0.89	0.06		0.06 - 0.06
7	16	0.0	89984	-0.91	0.10		0.10 - 0.10
8	4234	4.7	85766	-0.61	0.09		0.09 - 0.09

Record# 28 Start - End Times 4:10 - 4:24 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	66	0.1	89934	-0.02	0.00		0.00 - 0.00
2	2984	3.3	87016	-0.20	0.00		0.00 - 0.00
3	40	0.0	89960	-0.16	0.01		0.01 - 0.01
5	1582	1.8	88418	-0.89	0.40		0.39 - 0.41
7	2694	3.0	87306	-0.92	0.53		0.53 - 0.56
8	12326	13.7	77674	-0.60	0.54		0.50 - 0.64

Record# 29 Start - End Times 4:25 - 4:39 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	66	0.1	89934	-0.02	0.00		0.00 - 0.00
2	275	0.3	89725	-0.20	0.00		0.00 - 0.00
3	19	0.0	89981	-0.16	0.00		0.00 - 0.00
5	54	0.1	89946	-0.89	0.16		0.16 - 0.16
7	23	0.0	89977	-0.93	0.33		0.33 - 0.33
8	35659	39.6	54341	-0.59	0.16		0.13 - 0.24

Record# 30 Start - End Times 4:40 - 4:54 Count = 90000

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER
1	79	0.1	89921	-0.02	0.00		0.00 - 0.00
2	75	0.1	89925	-0.20	0.00		0.00 - 0.00
3	19	0.0	89981	-0.16	0.00		0.00 - 0.00
5	248	0.3	89752	-0.89	0.04		0.04 - 0.04
7	23	0.0	89977	-0.89	0.04		0.04 - 0.04
8	21767	24.2	68233	-0.59	0.03		0.03 - 0.04

Record# 31 Start - End Times 4:55 - 0: 0 Count = 83360

CHANNEL	ERRORS	%ERRORS	SAMPLES	MEAN	STAND. DEV.	CONFIDENCE INTERVAL:	LOWER - UPPER

1	200	0.2	83160	-0.02	0.04	0.04 - 0.04
2	11281	13.5	72079	-0.20	0.00	0.00 - 0.00
3	88	0.1	83272	-0.16	0.03	0.03 - 0.03
5	417	0.5	82943	-0.91	1.14	1.14 - 1.15
7	207	0.2	83153	-0.91	1.17	1.17 - 1.17
8	5820	7.0	77540	-0.62	1.21	1.17 - 1.33

Appendix C

Activity Indices Plots

Below are the activity index plots for the Spacelab D1 mission. The data for these can be found in the "Raw D1PREPHG Output" Appendix.

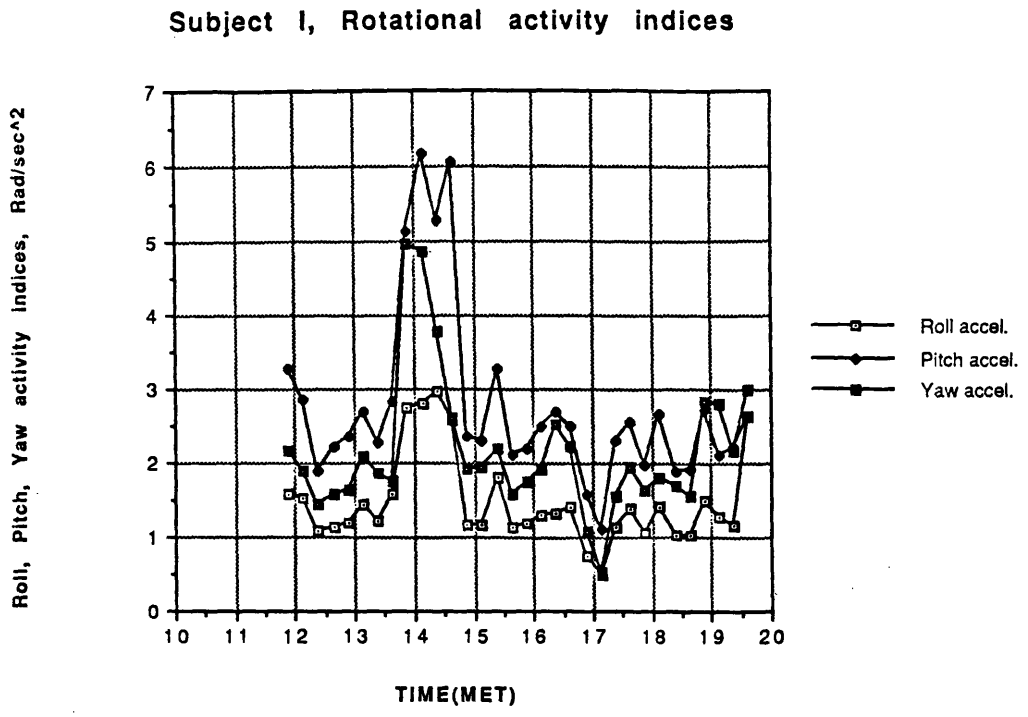

Figure C.1: Angular activity indices for subject I

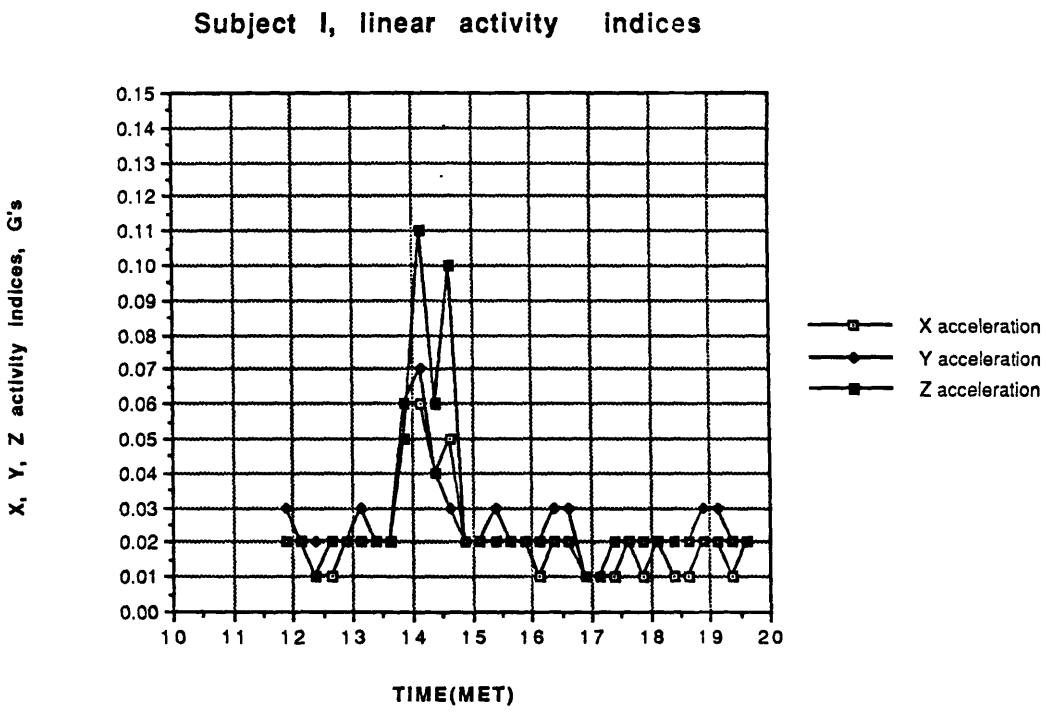

Figure C.2: Linear activity indices for subject I

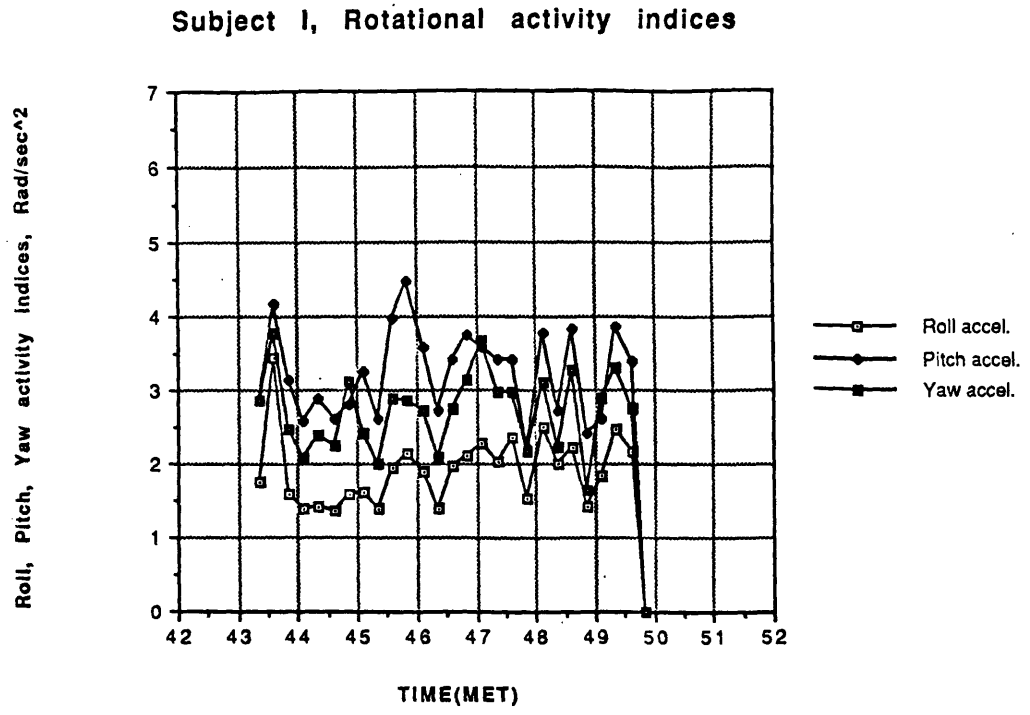

Figure C.3: Angular activity indices for subject I

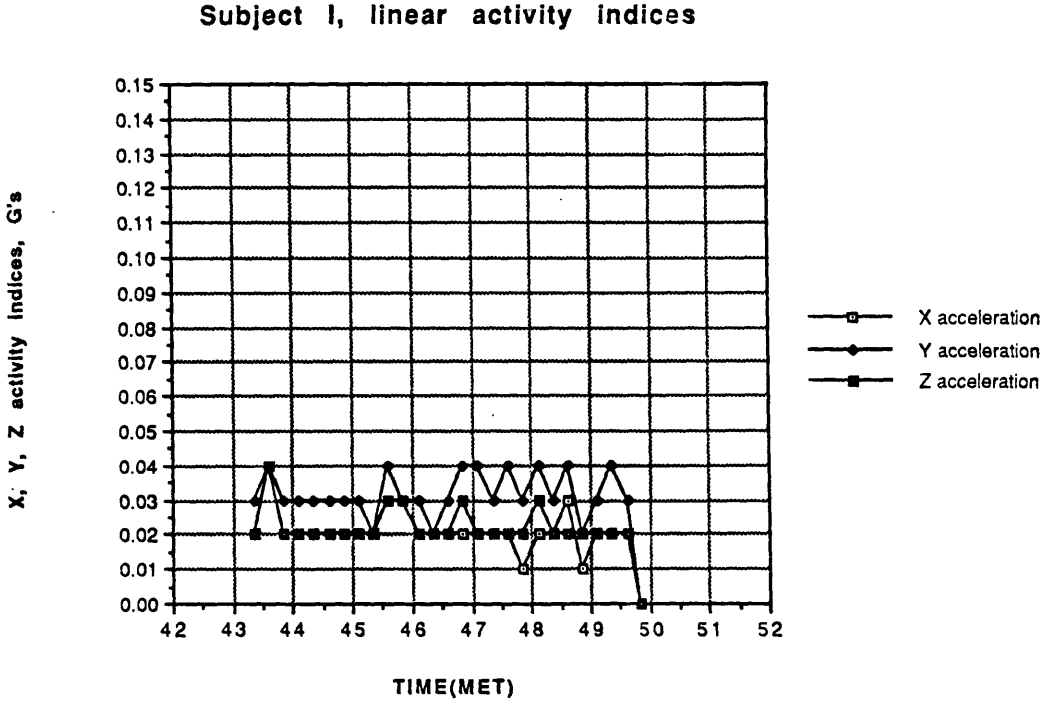

Figure C.4: Linear activity indices for subject I

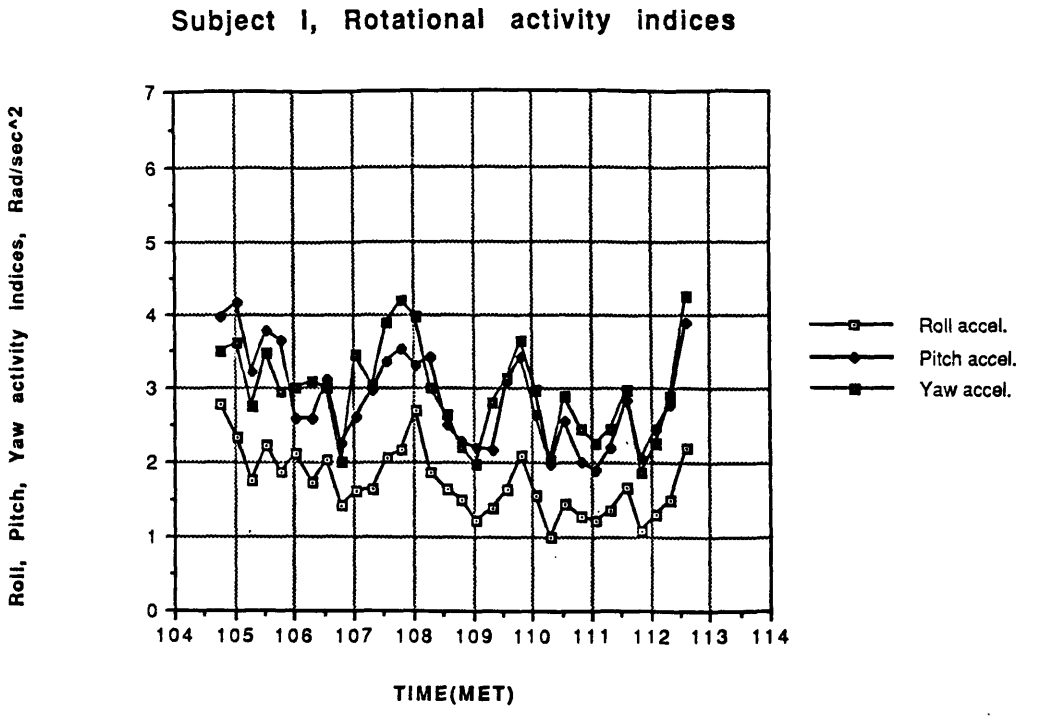

Figure C.5: Angular activity indices for subject I

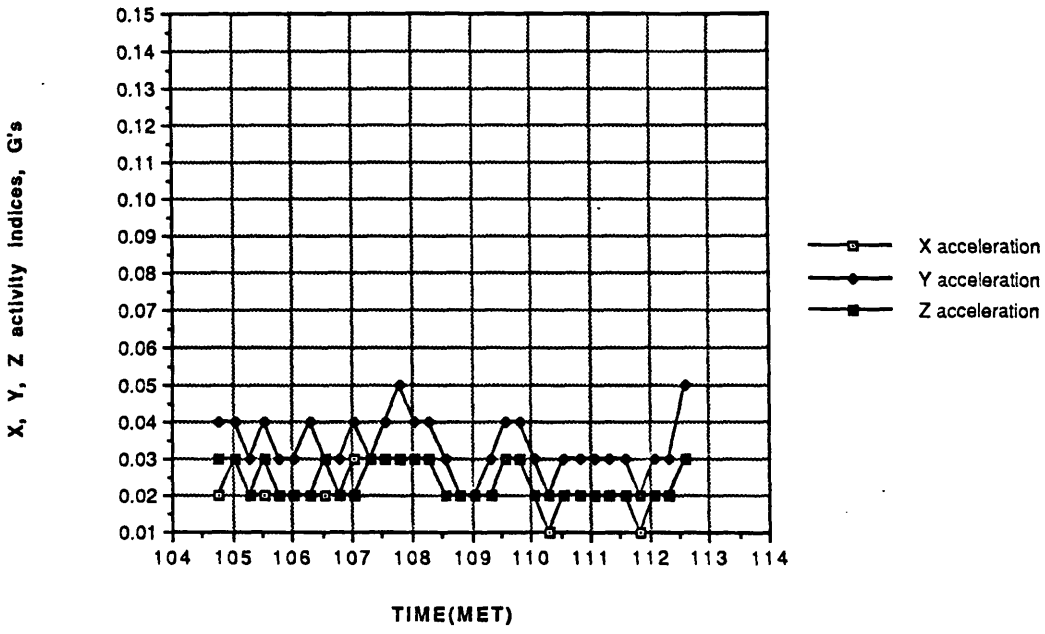

Figure C.6: Linear activity indices for subject I

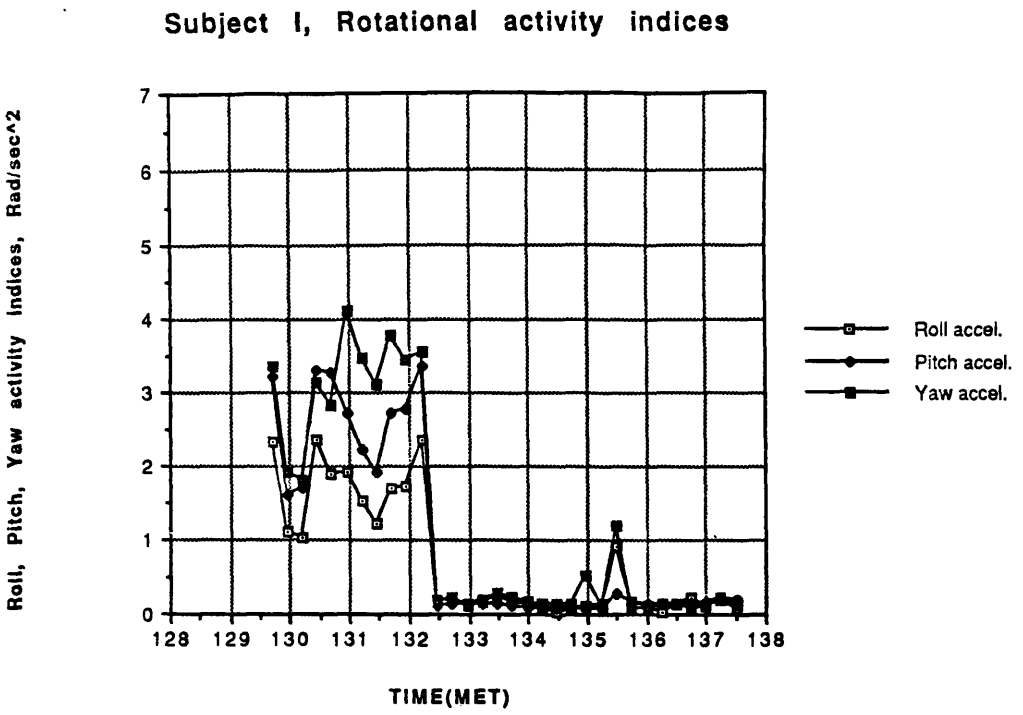

Figure C.7: Angular activity indices for subject I

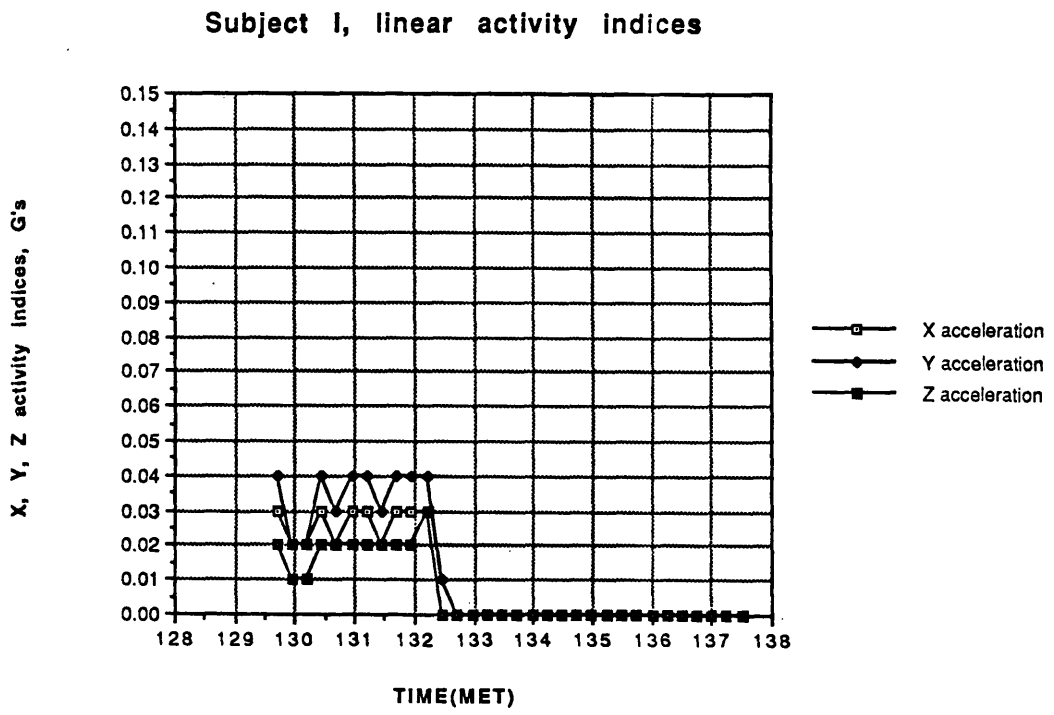

Figure C.8: Linear activity indices for subject I

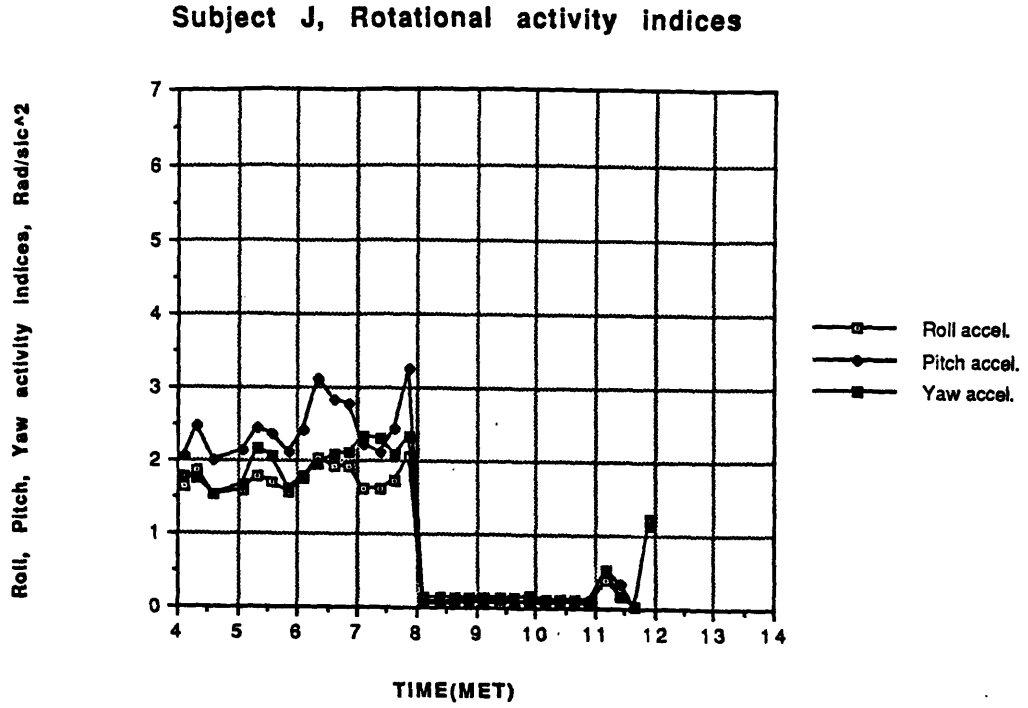

Figure C.9: Angular activity indices for subject J

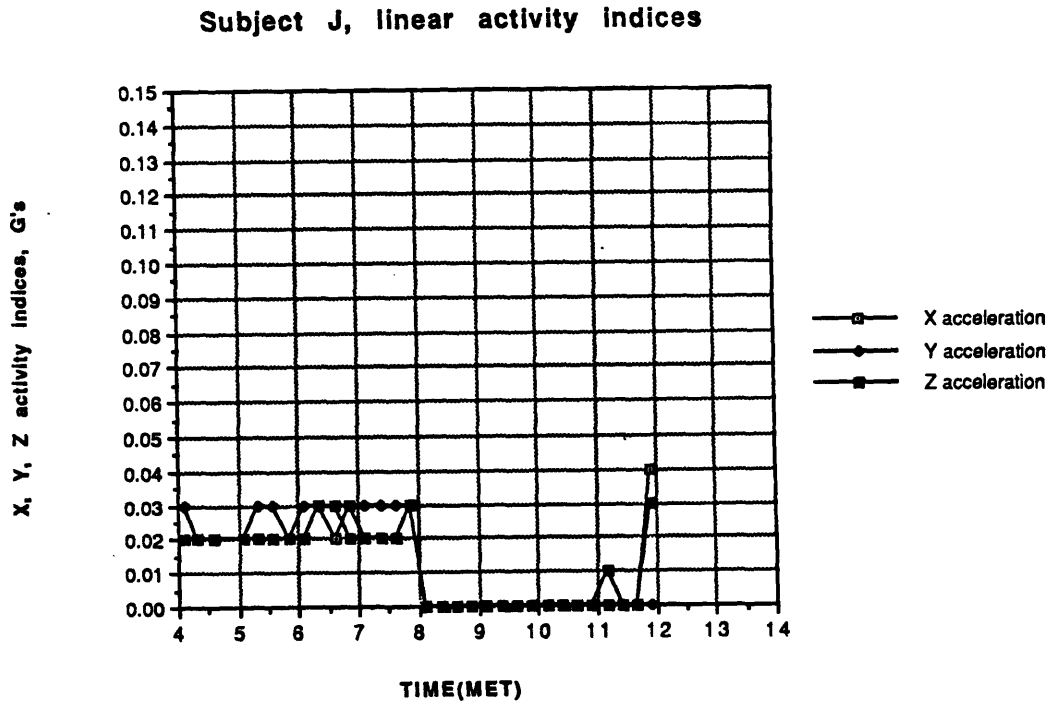

Figure C.10: Linear activity indices for subject J

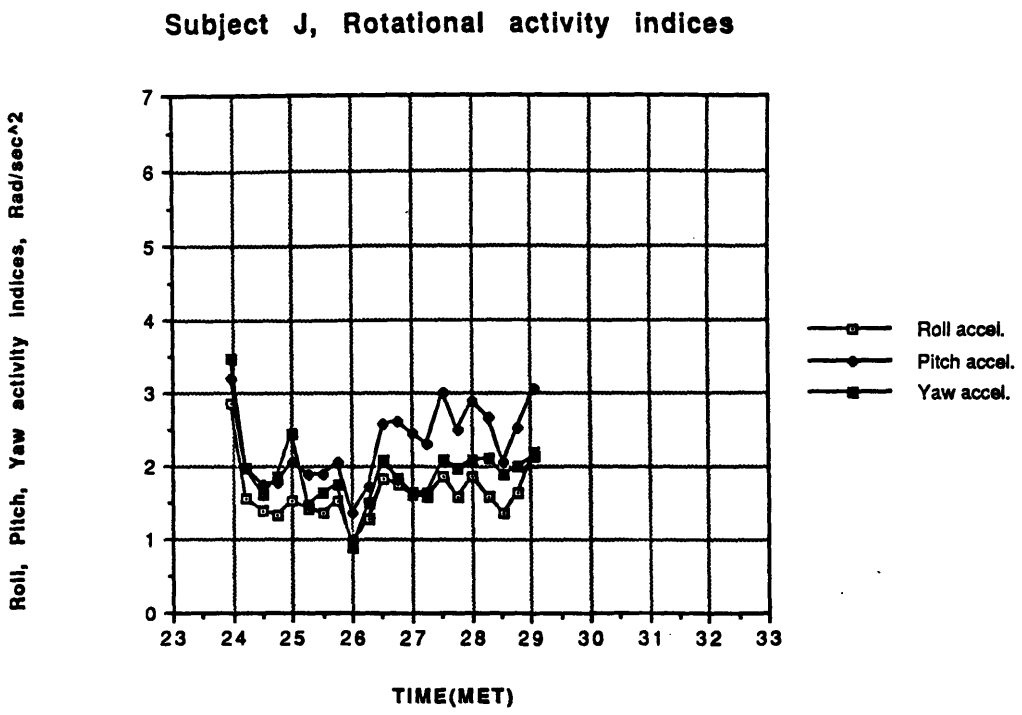

Figure C.11: Angular activity indices for subject J

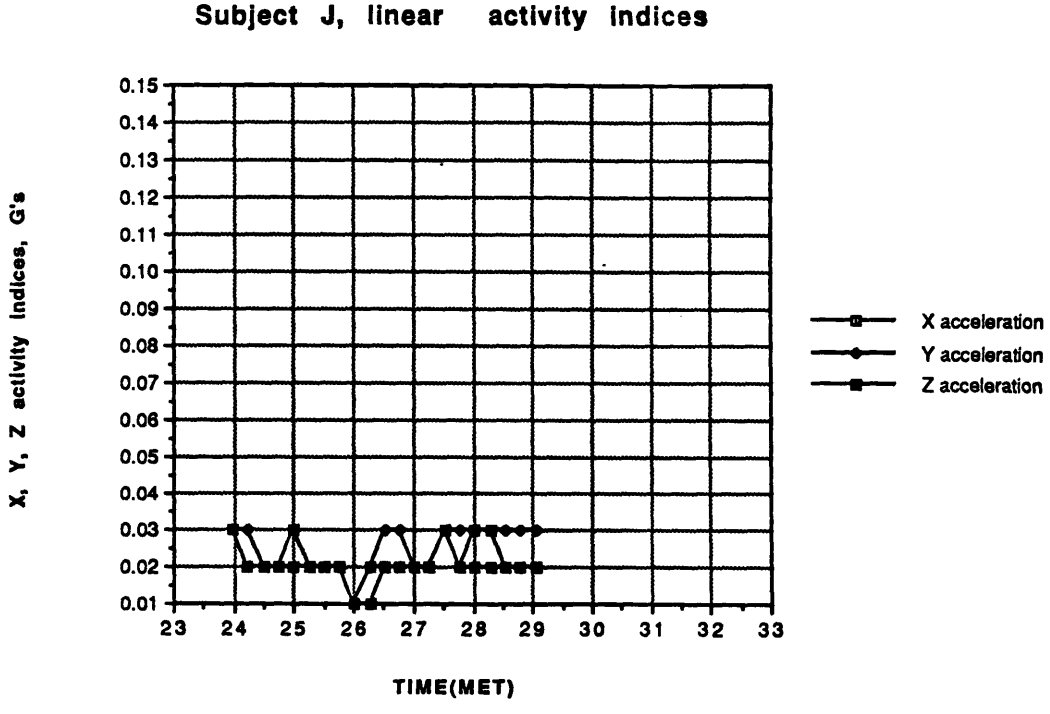

Figure C.12: Linear activity indices for subject J

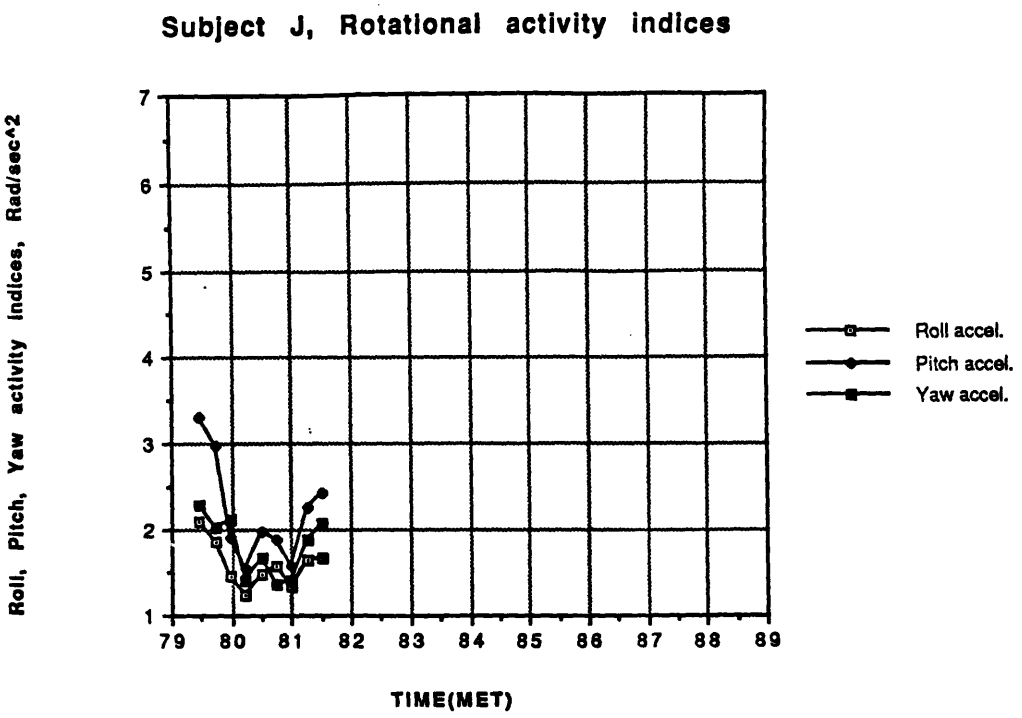

Figure C.13: Angular activity indices for subject J

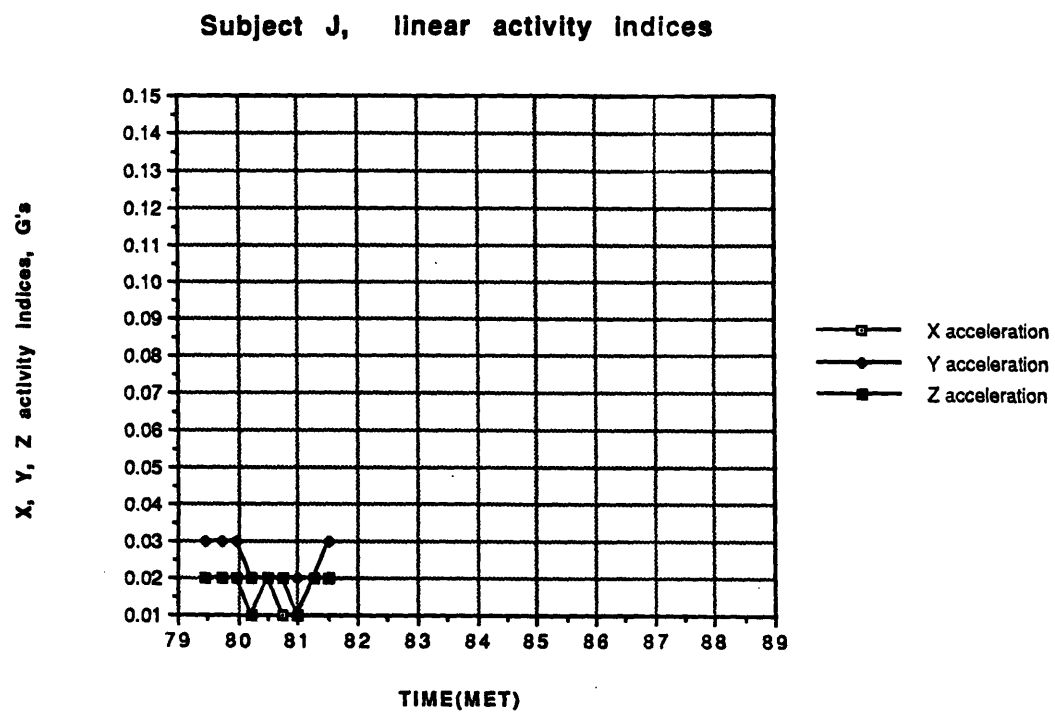

Figure C.14: Linear activity indices for subject J

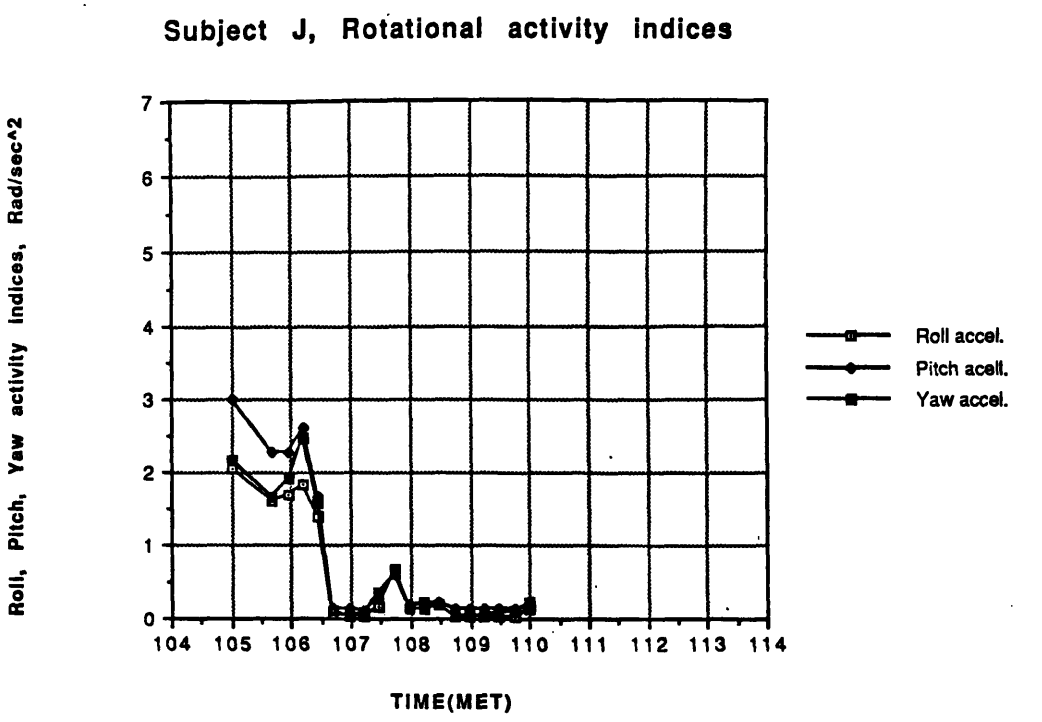

Figure C.15: Angular activity indices for subject J

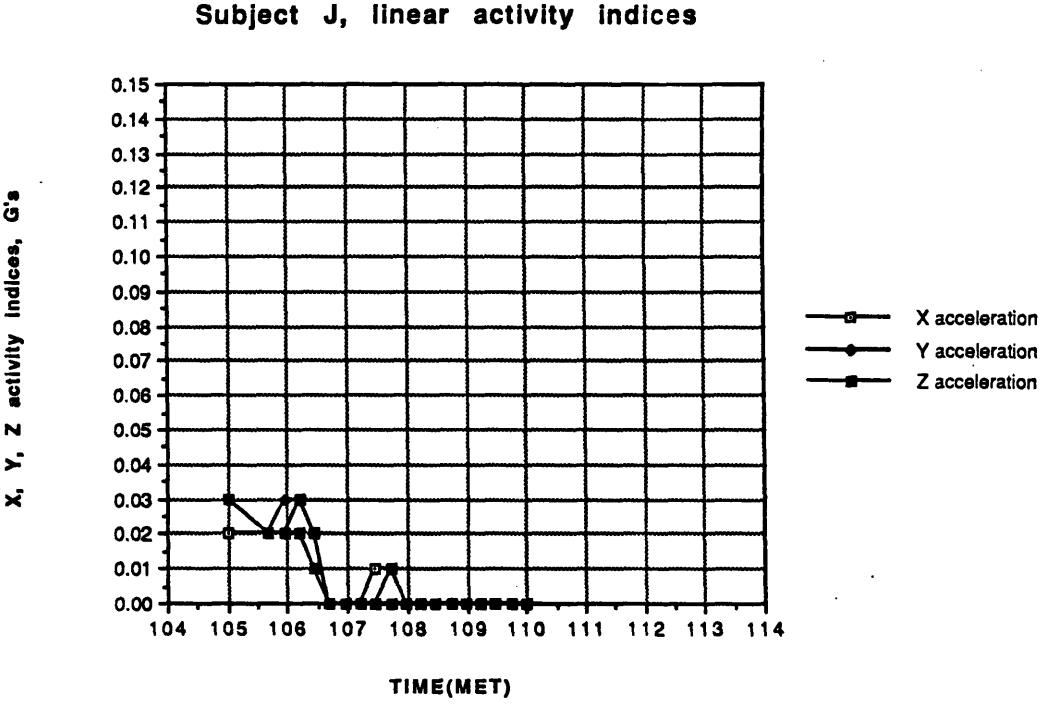

Figure C.16: Linear activity indices for subject J

Appendix D

File Index

D.1 Spacelab D1 Mission Files

The D1 mission files used in this thesis were:

- D1SN1091.DAT
- D1SN1093.DAT
- D1SN1094.DAT
- D1SN1109.DAT
- D1SN1110.DAT
- D1SN1114.DAT
- D1SN1125.DAT

The numbers in the name of these files correspond to the CDTR tape serial number.

D.2 Spacelab 1 Mission Files

The SL1 mission files used in this thesis were:

- SN1001.SL1

- SN1002.SL1
- SN1003.SL1
- SN1005.SL1
- SN1006.SL1
- SN1007.SL1
- SN1008.SL1
- SN1009.SL1
- SN1012.SL1
- SN1014.SL1
- SN1015.SL1
- SN1016.SL1
- SN1017.SL1
- SN1027.SL1
- SN1028.SL1
- SN1029.SL1
- SN1030.SL1
- SN1032.SL1

The following files were not used due to high error rates:

- SN1004.SL1

- SN1011.SL1

File SN1004.SL1 was not used because channel 7 (pitch) contained less than 1000 valid data points during most of the 15 minute intervals. Normally, there should be about 90,000 points in a 15 minute interval. File SN1011.SL1 was not used because channel 8 (yaw) contained no valid data at all in most of the 15 minute intervals. The total number of point in this channel was about 15,000.

The SN1013.SL1 file was not used because it appeared that the ARU was not worn during this period, or there was some other problem.

The numbers in the name of these files correspond to the CDTR tape serial number.

The MET times for these files were taken from McCoy's thesis.

Appendix E

Error_cleanup and Its Subroutines

E.1 Error_cleanup

```
      PROGRAM ERROR_CLEANUP
C     MAIN PROGRAM TO CLEAN UP THE ERRORS FROM DATA FILES

C     ------------------------------------------------

C     VARIABLE TO HOLD THREE RECORDS OF CURRENTLY READ DATA
      LOGICAL*2 DATA_BLOCK (8,320,3)/7680*0/

      LOGICAL*1 KEEP_READING /.TRUE./ ! A LOCAL UTILITY VARIABLE

      LOGICAL*1 TIME_CODE_FLAG

      LOGICAL*1 CLEAN_SMOOTH_FLAG

C     A LOCAL UTILITY COUNTER USED TO INPUT NUMBERS OF CHANNELS
C     TO BE PROCESSED.
      INTEGER*2 COUNTER1

C     IMPORTANT COUNTER USED FOR LOOPING THROUGH ALL CHANNELS
C     DURING ERROR PROCESSING.
      INTEGER*2 CHANNEL_COUNTER1

C     A UTILITY VARIABLE USED IN THE IMPLIED DO LOOPS DURING
C     READING OF THE DATA INTO DATA_BLOCK ARRAY.
      INTEGER*2 CHANNEL_COUNTER2

C     LOCAL UTILITY VARIABLE USED DURING INPUT OF CHANNEL
C     NUMBERS TO BE PROCESSED.
      INTEGER*2 ACCEPT_CHANNEL /-1/

C     ARRAY THAT SHOWS WHICH CHANNELS WERE CHOSEN FOR PROCESSING.
C     USED IN THE MAIN LOOP TO DECIDE WHETHER TO PROCESS DATA
C     FROM THAT CHANNEL OR NOT.
      INTEGER*2 DATA_IN_CHANNEL(8)/8*0/

C     A LOCAL UTILITY VARIABLE USED TO COUNT HOW MANY CHANNELS WERE
C     ENTERED FOR PROCESSING.
      INTEGER*2 CHANNELS_ENTERED /0/

C     VARIABLE USED TO HOLD THE GUESS FOR THE ERROR TO BE USED IN
C     REPORTING OF ERROR (NOT CORRECTING IT)
      INTEGER*2 ERROR1_GUESS
```

```
C       THE SAME AS ABOVE BUT FOR THE SECOND ERROR IN AN ERROR STRING
        INTEGER*2 ERROR2_GUESS

C       VARIABLE USED TO SPECIFY WHAT ACTION TO TAKE ON THE RETURNED
C       ERROR (NOT IDENTIFYING WHICH ERROR IT IS)
        INTEGER*2 CORRECTIVE_ACTION_CODE1

C       THE SAME AS ABOVE BUT FOR THE SECOND ERROR IN AN ERROR STRING
        INTEGER*2 CORRECTIVE_ACTION_CODE2

C       A PARAMETER DECLARATION. THIS PARAMETER IS USED WITH THE
C       DATA_IN_CHANNEL ARRAY TO DECIDE WHETHER TO PROCESS DATA IN
C       A CHANNEL OR NOT.
        INTEGER*2 NO_DATA

C       PARAMETER USED FOR STRIPPING EXTRA BITS OFF DATA
        INTEGER*2 DATA_MASK

C       A UTILITY VARIABLE USED TO READ IN AND WRITE OUT THE FIRST
C       THREE RECORDS OF DATA.
        INTEGER*2 RECORD_COUNTER1

C       VARIABLE SHOWING WHICH PART OF THE DATA_BLOCK ARRAY CONTAINS
C       RECORD PRESENTLY BEING PROCESSED.
        INTEGER*2 PRESENT_RECORD_INDEX /2/

C       AN ARRAY USED TO "INCREMENT" THE PRESENT_RECORD_INDEX
        INTEGER*2 NEXT_INDEX(3) /2,3,1/

C       AN ARRAY USED TO "DECREMENT" THE PRESENT_RECORD_INDEX
        INTEGER*2 PREVIOUS_INDEX(3) /3,1,2/

C       IMPORTANT COUNTER USED FOR LOOPING THROUGH ALL THE SAMPLES
C       DURING ERROR PROCESSING
        INTEGER*2 SAMPLE_COUNTER1

C       A UTILITY VARIABLE USED IN THE IMPLIED DO LOOPS DURING
C       READING OF THE DATA INTO DATA_BLOCK ARRAY.
        INTEGER*2 SAMPLE_COUNTER2

C       A UTILITY COUNTER USED IN READING IN THE EXPECTED AVERAGES
C       OF DATA IN DIFFERENT CHANNELS
        INTEGER*2 AVERAGE_COUNTER2

C       UTILITY COUNTER
        INTEGER*2 CLEANUP_COUNTER1

C       VARIABLE USED FOR HOLDING A DATA VALUE THAT MAY BE USED
C       DURING ERROR CORRECTION
        INTEGER*2 CORRECTION_VALUE1

C       THE SAME AS ABOVE BUT FOR THE SECOND ERROR IN THE ERROR STRING
        INTEGER*2 CORRECTION_VALUE2

C       VARIABLE HOLDING SAMPLE NUMBER OF THE BEGINNING OF THE
C       FIRST ERROR IN ERROR STRING.
        INTEGER*2 START_OF_ERROR1_SAMPLE

C       THE SAME AS ABOVE BUT FOR THE END OF FIRST ERROR IN STRING.
        INTEGER*2 END_OF_ERROR1_SAMPLE

C       THE SAME AS ABOVE BUT FOR THE START OF SECOND ERROR IN ERROR
C       STRING.
        INTEGER*2 START_OF_ERROR2_SAMPLE
```

```
C       THE SAME AS ABOVE BUT FOR THE END OF SECOND ERROR IN STRING.
        INTEGER*2 END_OF_ERROR2_SAMPLE

        INTEGER*2 TIME_CODE

        INTEGER*2 FLUSH_OUT

        INTEGER*2 START_UNMARKED_ERROR

        INTEGER*2 MAXIMUM_SMOOTH_JUMP(8)/3*15,10,60,20,2*60/

        INTEGER*2 TIME_BLOCK_COUNTER /1/

        INTEGER*2 PRINT_OUT_COUNTER1,PRINT_OUT_COUNTER2,
        2         PRINT_OUT_COUNTER3,SUMMARY_COUNTER1,
        2         SUMMARY_COUNTER2,SUMMARY_COUNTER3,LOCAL_COUNTER50,
        2         LOCAL_COUNTER51

        INTEGER*2 CURRENT_CLEAN_VALUE,
        2         PREVIOUS_CLEAN_VALUE

C       VARIABLE TO HOLD THE NUMBER OF THE CURRENTLY PROCESSED RECORD
        INTEGER*4 RECORD_NUMBER

C       VARIABLE THAT HOLDS THE NUMBER OF THE FIRST RECORD IN THE INPUT
C       FILE TO BE PROCESSED (MINIMUM IS 2)
        INTEGER*4 FIRST_RECORD

C       VARIABLE THAT HOLDS THE NUMBER OF THE LAST RECORD IN THE INPUT
C       FILE TO BE PROCESSED (MINIMUM IS FIRST_RECORD)
        INTEGER*4 LAST_RECORD

C       THE NEXT FOUR VARIABLES HOLD RECORD NUMBERS OF THE BEGINNINGS
C       AND ENDS OF THE FIRST AND SECOND ERRORS IN ERROR STRING.
        INTEGER*4 START_OF_ERROR1_REC,
        2         END_OF_ERROR1_REC,
        2         START_OF_ERROR2_REC,
        2         END_OF_ERROR2_REC

        INTEGER*4 ERROR_STATISTICS(8,17,100)/13600*0/
        INTEGER*4 CLEAN_STATISTICS(8,17,100)/13600*0/
        INTEGER*4 ERROR_LENGTH_STATISTICS(8,17,100)/13600*0/
        INTEGER*4 CLEAN_LENGTH_STATISTICS(8,17,100)/13600*0/
        INTEGER*4 SUMMARY_STATISTICS(8,17)/136*0/
        INTEGER*4 CLEAN_SUMMARY_STATISTICS(8,17)/136*0/
        INTEGER*4 SUMMARY_LENGTH_STATISTICS(8,17)/136*0/
        INTEGER*4 CLEAN_SUMMARY_LENGTH_STATISTICS(8,17)/136*0/
        INTEGER*4 COMBINED_BLOCK_ERRORS(8)/8*0/
        INTEGER*4 COMBINED_BLOCK_LENGTH(8)/8*0/
        INTEGER*4 COMBINED_CLEAN_ERRORS(8)/8*0/
        INTEGER*4 COMBINED_CLEAN_LENGTH(8)/8*0/

        REAL*8  SUMMARY_PERCENTAGE (8,17)/136*0.0/
        REAL*8  SUMMARY_CLEAN_PERCENTAGE (8,17)/136*0.0/


        INTEGER*4 TIME_SAMPLE_COUNTER /0/

C       VARIABLE TO HOLD THE APPROXIMATION TO THE AVERAGE OF THE
C       VALID DATA IN EACH CHANNEL
        REAL*4 AVERAGE(8)/512.0,512.0,512.0,512.0,512.0,512.0,
        2       510.0,512.0/
```

```
      REAL*4 SMOOTHNESS_CRITERIA(8)/3*16.0,3.0,400.0,16.0,2*400.0/

C     ''ADJUSTED'' MEANS THAT THEY ARE DIVIDED BY 100 SO THAT
C     WHEN USED IN FORMULAS BELOW THEY WILL GIVE PERCENTAGES
C     RATHER THAN FRACTIONS

      REAL*8 ADJUSTED_TOTAL_LENGTH /3.2/
      REAL*8 ADJUSTED_TIME_BLOCK_LENGTH /900.0/

C     THE NEXT THREE VARIABLES HOLD THE NAMES OF INPUT,OUTPUT, AND
C     REPORT FILES.
      CHARACTER*50 INPUT_FILE,OUTPUT_FILE,REPORT_FILE

C     SEE DECLARATION FOR EIPLANATIONS
      PARAMETER (TIME_CODE =0)
      PARAMETER (NO_DATA = 0)
      PARAMETER (DATA_MASK = 1023)
      PARAMETER (FLUSH_OUT = 10)
      PARAMETER (START_UNMARKED_ERROR =11)
      COMMON /BLOCK1/ SMOOTHNESS_CRITERIA,
     2                MAXIMUM_SMOOTH_JUMP

C     ------------------------------------------------

C     ------------------------------------------------

C     FORMAT STATEMENTS

10    FORMAT(/,' PLEASE ENTER DATA FILE SPECIFICATIONS: ',$)
11    FORMAT(/,' ERROR GUESS ',I2,' CHANNEL ',I2,
     2       ' NUMBER OF TIMES ',I8,' TOTAL LENGTH ',I8,
     2       ' PERCENTAGE ',F8.4)
12    FORMAT(/,' PLEASE ENTER EXPECTED AVERAGE FOR CHANNEL ',I2)
14    FORMAT(F6.0)
16    FORMAT(/,' FLUSHING OUT ERROR IN CHANNEL ',I2,' RECORD ',
     2       I7,' SAMPLE ',I3)
17    FORMAT(/,' PRINTING ERROR STATISTICS FOR 15 MIN BLOCK
     2NUMBER ',I4)
18    FORMAT(/,' ERROR GUESS ',I2,' CHANNEL ',I2,
     2       ' NUMBER OF TIMES ',I8,' TOTAL LENGTH ',I8,
     2       ' PERCENTAGE ',F8.4)
19    FORMAT(/,' ERRORS IN INPUT DATA FILE ')
20    FORMAT(1A)
21    FORMAT(/,' ERRORS IN CORRECTED FILE ')
22    FORMAT(/,' SUMMARY STATISTICS ')
23    FORMAT(/,' UNMARKED DATA NOT SMOOTH IN CHANNEL ',I2,
     2' RECORD ',I8,' SAMPLE ',I3)
24    FORMAT(/,' CHANNEL ',I1,
     2         ' NUMBER OF COMBINED ERRORS IN CHANNEL ',I5,
     2         ' TOTAL LENGTH OF ERRORS IN CHANNEL ',I5)
30    FORMAT(/,' PLEASE ENTER OUTPUT FILE SPECIFICATIONS: ',$)
40    FORMAT(/,' PLEASE ENTER REPORT FILE SPECIFICATIONS: ',$)
50    FORMAT(/,' PLEASE ENTER THE CHANNEL TO BE PROCESSED.',
     2/,' ENTER -1 TO EXIT SELECTION, 0 TO SELECT ALL CHANNELS.')
60    FORMAT(/,' PLEASE ENTER THE NEXT CHANNEL: ',$)
70    FORMAT(BN,I2)
77    FORMAT(/,' ERROR GUESS ',I2,' CHANNEL ',I2,
     2         ' START REC ',I5,' SAMPLE ',I3,' END REC ',I5,
     2         ' SAMPLE ',I3)
80    FORMAT(BN,I12)
90    FORMAT(/,' PLEASE ENTER THE FIRST RECORD TO BE PROCESSED: ',$)
91    FORMAT(/,' PLEASE ENTER THE LAST RECORD TO BE PROCESSED: ',$)
```

```
C       ------------------------------------------------

C       ------------------------------------------------

C       READ THE NAME OF THE INPUT FILE:

C       DO WHILE (INPUT_FILE .NE. '')
           WRITE (6,10)
           READ  (5,20) INPUT_FILE
C       END DO

C       OPEN INPUT FILE FOR READING

        OPEN  (UNIT=1,ACCESS='DIRECT',FORM='UNFORMATTED',
        2 STATUS='OLD',READONLY,FILE=INPUT_FILE)

C       READ THE NAME OF THE OUTPUT FILE:

C       DO WHILE (OUTPUT_FILE .NE. '')
           WRITE (6,30)
           READ  (5,20) OUTPUT_FILE
C       END DO

C       OPEN NEW OUTPUT FILE FOR WRITING

        OPEN   (UNIT=2,ACCESS='DIRECT',FORM='UNFORMATTED',
        2 STATUS='NEW',RECL=1280,FILE=OUTPUT_FILE)

C       READ THE NAME OF THE REPORT FILE:

C       DO WHILE (REPORT_FILE .NE. '')
           WRITE (6,40)
           READ  (5,20) REPORT_FILE
C       END DO

C       OPEN NEW REPORT FILE FOR WRITING

        OPEN    (UNIT=3,STATUS='NEW',FILE=REPORT_FILE)

C       ------------------------------------------------

C       ------------------------------------------------
C       READ THE NUMBERS OF THE FIRST AND LAST RECORDS TO BE PROCESSED.

        FIRST_RECORD=2
        LAST_RECORD=2

        WRITE (6,90)
        READ  (5,80) FIRST_RECORD

        DO WHILE (FIRST_RECORD .LE. 2)
           WRITE (6,90)
           READ  (5,80) FIRST_RECORD
        END DO

        WRITE(6,91)
        READ (5,80)LAST_RECORD

        DO WHILE (LAST_RECORD .LT. FIRST_RECORD)
           WRITE(6,91)
           READ (5,80)LAST_RECORD
        END DO
```

```
C       READ WHICH CHANNELS TO PROCESS

        WRITE(6,50)
        DO WHILE (KEEP_READING)
           WRITE(6,60)
           READ(5,70)ACCEPT_CHANNEL

           IF ((ACCEPT_CHANNEL .EQ. -1).AND.
     2     (CHANNELS_ENTERED .NE. 0))THEN
              KEEP_READING=.FALSE.
           END IF


           IF (ACCEPT_CHANNEL .EQ. 0)THEN
              DO COUNTER1=1,8,1
                 DATA_IN_CHANNEL(COUNTER1)=1
              END DO
              KEEP_READING=.FALSE.
           END IF

           IF ((ACCEPT_CHANNEL .GE. 1).AND.
     2     (ACCEPT_CHANNEL .LE. 8)) THEN
              CHANNELS_ENTERED=CHANNELS_ENTERED+1-DATA_IN_CHANNEL
     2        (ACCEPT_CHANNEL)
              DATA_IN_CHANNEL(ACCEPT_CHANNEL)=1
           END IF
        END DO

        DO AVERAGE_COUNTER2=1,8,1
           IF(DATA_IN_CHANNEL(AVERAGE_COUNTER2) .NE. NO_DATA)THEN
              WRITE (6,12) AVERAGE_COUNTER2
              READ  (5,14) AVERAGE(AVERAGE_COUNTER2)
           END IF
        END DO

C       ---------------------------------------------

C       ---------------------------------------------


C       READ AND WRITE ''FIRST'' THREE RECORDS OF DATA:

        RECORD_NUMBER=FIRST_RECORD

        DO RECORD_COUNTER1=-1,1,1
           READ (1,REC=(RECORD_NUMBER+RECORD_COUNTER1),ERR=1000)
     2        ((DATA_BLOCK(CHANNEL_COUNTER2,SAMPLE_COUNTER2,
     2                     RECORD_COUNTER1+2),
     2          CHANNEL_COUNTER2=1,8,1),
     2          SAMPLE_COUNTER2=1,320,1)
           WRITE(2,REC=(RECORD_COUNTER1+2),
     2           ERR=2000)
     2        ((DATA_BLOCK(CHANNEL_COUNTER2,SAMPLE_COUNTER2,
     2                     RECORD_COUNTER1+2),
     2          CHANNEL_COUNTER2=1,8,1),
     2          SAMPLE_COUNTER2=1,320,1)
        END DO

C       ---------------------------------------------

C       ---------------------------------------------

C       MAIN LOOP OF THE PROGRAM
```

```
100     DO SAMPLE_COUNTER1=1,320,1 !LOOPS THROUGH ALL 320
C                                  SAMPLES IN EVERY RECORD

        TIME_SAMPLE_COUNTER=TIME_SAMPLE_COUNTER+1
        IF (TIME_SAMPLE_COUNTER .EQ. 90000)THEN
           TIME_SAMPLE_COUNTER=0
           TIME_BLOCK_COUNTER=TIME_BLOCK_COUNTER+1
        END IF

        CALL TIME_CODE_CHECK(DATA_BLOCK(1,SAMPLE_COUNTER1,
        2                               PRESENT_RECORD_INDEX),
        2                    TIME_CODE_FLAG)
        IF(TIME_CODE_FLAG)THEN
           GO TO 350
        END IF

200        DO CHANNEL_COUNTER1=1,8,1 !LOOPS THROUGH ALL 8
C                                    CHANNELS
C             ----------------------------------------
C             THIS SUBROUTINE
C             CHECKS TO SEE IF THERE IS AN ERROR IN THE CHANNEL,
C             AND CALLS THE ERROR I.D. SUBROUTINE, IF NEEDED.

              IF(DATA_IN_CHANNEL(CHANNEL_COUNTER1) .EQ. NO_DATA) THEN
                 GO TO 300
              END IF

202           CONTINUE

              CORRECTION_VALUE1=0
              CALL FOR_EACH_CHANNEL (CHANNEL_COUNTER1,
        2                            RECORD_NUMBER,
        2                            SAMPLE_COUNTER1,
        2                            PRESENT_RECORD_INDEX,
        2                            FIRST_RECORD,
        2                            DATA_BLOCK,
        2                            AVERAGE(CHANNEL_COUNTER1),
        2                            START_OF_ERROR1_REC,
        2                            START_OF_ERROR1_SAMPLE,
        2                            END_OF_ERROR1_REC,
        2                            END_OF_ERROR1_SAMPLE,
        2                            START_OF_ERROR2_REC,
        2                            START_OF_ERROR2_SAMPLE,
        2                            END_OF_ERROR2_REC,
        2                            END_OF_ERROR2_SAMPLE,
        2                            ERROR1_GUESS,
        2                            ERROR2_GUESS,
        2                            CORRECTIVE_ACTION_CODE1,
        2                            CORRECTION_VALUE1,
        2                            CORRECTIVE_ACTION_CODE2,
        2                            CORRECTION_VALUE2)
C             ----------------------------------------

C       ------------------------------------------------------

C             CLEAN UP THE FIRST ERROR

              GO TO (205,210,215,220,225,230,235,240,245),
        2           CORRECTIVE_ACTION_CODE1

C             AT 205 - DO NOTHING
205           CONTINUE

              GO TO 250
```

```
C            AT 210 - CLEAR THE ERROR FLAG AND DO NOTHING ELSE
210          CONTINUE
             CALL CLEAR_ERROR_FLAG ((START_OF_ERROR1_REC
         2                           -FIRST_RECORD+2),
         2                          (END_OF_ERROR1_REC
         2                           -FIRST_RECORD+2),
         2                          START_OF_ERROR1_SAMPLE,
         2                          END_OF_ERROR1_SAMPLE,
         2                          CHANNEL_COUNTER1)
             CALL UPDATE_DATA_BLOCK (DATA_BLOCK,
         2                           PRESENT_RECORD_INDEX,
         2                           (RECORD_NUMBER
         2                            -FIRST_RECORD+2),
         2                           (START_OF_ERROR1_REC
         2                            -FIRST_RECORD+2),
         2                           (END_OF_ERROR1_REC
         2                            -FIRST_RECORD+2))


             WRITE(3,77)ERROR1_GUESS,CHANNEL_COUNTER1,
         2              START_OF_ERROR1_REC,
         2              START_OF_ERROR1_SAMPLE,END_OF_ERROR1_REC,
         2              END_OF_ERROR1_SAMPLE

             CALL UPDATE_ERROR_STATISTICS(ERROR_STATISTICS,
         2                                ERROR_LENGTH_STATISTICS,
         2                                ERROR1_GUESS,
         2                                START_OF_ERROR1_REC,
         2                                END_OF_ERROR1_REC,
         2                                START_OF_ERROR1_SAMPLE,
         2                                END_OF_ERROR1_SAMPLE,
         2                                CHANNEL_COUNTER1,
         2                                TIME_BLOCK_COUNTER,
         2                                TIME_SAMPLE_COUNTER)

             GO TO 250

C            AT 215 - SET VALUE TO CORRECTION VALUE
C                     AND CLEAR THE ERROR FLAG
215          CONTINUE

             CALL SET_TO_VALUE (CORRECTION_VALUE1,
         2                      (START_OF_ERROR1_REC
         2                           -FIRST_RECORD+2),
         2                      (END_OF_ERROR1_REC
         2                           -FIRST_RECORD+2),
         2                      START_OF_ERROR1_SAMPLE,
         2                      END_OF_ERROR1_SAMPLE,
         2                      CHANNEL_COUNTER1)

             CALL CLEAR_ERROR_FLAG ((START_OF_ERROR1_REC
         2                           -FIRST_RECORD+2),
         2                          (END_OF_ERROR1_REC
         2                           -FIRST_RECORD+2),
         2                          START_OF_ERROR1_SAMPLE,
         2                          END_OF_ERROR1_SAMPLE,
         2                          CHANNEL_COUNTER1)
             CALL UPDATE_DATA_BLOCK (DATA_BLOCK,
         2                           PRESENT_RECORD_INDEX,
         2                           (RECORD_NUMBER
         2                            -FIRST_RECORD+2),
         2                           (START_OF_ERROR1_REC
         2                            -FIRST_RECORD+2),
```

```
     2                          (END_OF_ERROR1_REC
     2                           -FIRST_RECORD+2))

          WRITE(3,77)ERROR1_GUESS,CHANNEL_COUNTER1,
     2               START_OF_ERROR1_REC,
     2               START_OF_ERROR1_SAMPLE,END_OF_ERROR1_REC,
     2               END_OF_ERROR1_SAMPLE
          CALL UPDATE_ERROR_STATISTICS(ERROR_STATISTICS,
     2                                 ERROR_LENGTH_STATISTICS,
     2                                 ERROR1_GUESS,
     2                                 START_OF_ERROR1_REC,
     2                                 END_OF_ERROR1_REC,
     2                                 START_OF_ERROR1_SAMPLE,
     2                                 END_OF_ERROR1_SAMPLE,
     2                                 CHANNEL_COUNTER1,
     2                                 TIME_BLOCK_COUNTER,
     2                                 TIME_SAMPLE_COUNTER)
          GO TO 250

C         AT 220 - SET THE ERROR FLAG
220       CONTINUE

          CALL SET_ERROR_FLAG ((START_OF_ERROR1_REC
     2                           -FIRST_RECORD+2),
     2                          (END_OF_ERROR1_REC
     2                           -FIRST_RECORD+2),
     2                          START_OF_ERROR1_SAMPLE,
     2                          END_OF_ERROR1_SAMPLE,
     2                          CHANNEL_COUNTER1)
          CALL UPDATE_DATA_BLOCK (DATA_BLOCK,
     2                            PRESENT_RECORD_INDEX,
     2                            (RECORD_NUMBER
     2                             -FIRST_RECORD+2),
     2                            (START_OF_ERROR1_REC
     2                             -FIRST_RECORD+2),
     2                            (END_OF_ERROR1_REC
     2                             -FIRST_RECORD+2))

          WRITE(3,77)ERROR1_GUESS,CHANNEL_COUNTER1,
     2               START_OF_ERROR1_REC,
     2               START_OF_ERROR1_SAMPLE,END_OF_ERROR1_REC,
     2               END_OF_ERROR1_SAMPLE
          CALL UPDATE_ERROR_STATISTICS(ERROR_STATISTICS,
     2                                 ERROR_LENGTH_STATISTICS,
     2                                 ERROR1_GUESS,
     2                                 START_OF_ERROR1_REC,
     2                                 END_OF_ERROR1_REC,
     2                                 START_OF_ERROR1_SAMPLE,
     2                                 END_OF_ERROR1_SAMPLE,
     2                                 CHANNEL_COUNTER1,
     2                                 TIME_BLOCK_COUNTER,
     2                                 TIME_SAMPLE_COUNTER)
          CALL UPDATE_ERROR_STATISTICS(CLEAN_STATISTICS,
     2                                 CLEAN_LENGTH_STATISTICS,
     2                                 ERROR1_GUESS,
     2                                 START_OF_ERROR1_REC,
     2                                 END_OF_ERROR1_REC,
     2                                 START_OF_ERROR1_SAMPLE,
     2                                 END_OF_ERROR1_SAMPLE,
     2                                 CHANNEL_COUNTER1,
     2                                 TIME_BLOCK_COUNTER,
     2                                 TIME_SAMPLE_COUNTER)
          GO TO 250
```

```
C           AT 225 - REPORT ONLY

225         CONTINUE

            WRITE(3,77)ERROR1_GUESS,CHANNEL_COUNTER1,
           2           START_OF_ERROR1_REC,
           2           START_OF_ERROR1_SAMPLE,END_OF_ERROR1_REC,
           2           END_OF_ERROR1_SAMPLE
            CALL UPDATE_ERROR_STATISTICS(ERROR_STATISTICS,
           2                             ERROR_LENGTH_STATISTICS,
           2                             ERROR1_GUESS,
           2                             START_OF_ERROR1_REC,
           2                             END_OF_ERROR1_REC,
           2                             START_OF_ERROR1_SAMPLE,
           2                             END_OF_ERROR1_SAMPLE,
           2                             CHANNEL_COUNTER1,
           2                             TIME_BLOCK_COUNTER,
           2                             TIME_SAMPLE_COUNTER)
            CALL UPDATE_ERROR_STATISTICS(CLEAN_STATISTICS,
           2                             CLEAN_LENGTH_STATISTICS,
           2                             ERROR1_GUESS,
           2                             START_OF_ERROR1_REC,
           2                             END_OF_ERROR1_REC,
           2                             START_OF_ERROR1_SAMPLE,
           2                             END_OF_ERROR1_SAMPLE,
           2                             CHANNEL_COUNTER1,
           2                             TIME_BLOCK_COUNTER,
           2                             TIME_SAMPLE_COUNTER)
            GO TO 250

C           AT 230 - RESCALE VALUE BY A MULTIPLE OF 2
230         CONTINUE

            CALL RESCALE_ERROR  ((START_OF_ERROR1_REC
           2                        -FIRST_RECORD+2),
           2                       (END_OF_ERROR1_REC
           2                        -FIRST_RECORD+2),
           2                       START_OF_ERROR1_SAMPLE,
           2                       END_OF_ERROR1_SAMPLE,
           2                       CHANNEL_COUNTER1,
           2                       CORRECTION_VALUE1,
           2                       SMOOTHNESS_CRITERIA
           2                         (CHANNEL_COUNTER1),
           2                       MAXIMUM_SMOOTH_JUMP
           2                         (CHANNEL_COUNTER1))

            CALL CLEAR_ERROR_FLAG ((START_OF_ERROR1_REC
           2                         -FIRST_RECORD+2),
           2                        (END_OF_ERROR1_REC
           2                         -FIRST_RECORD+2),
           2                        START_OF_ERROR1_SAMPLE,
           2                        END_OF_ERROR1_SAMPLE,
           2                        CHANNEL_COUNTER1)

            CALL UPDATE_DATA_BLOCK (DATA_BLOCK,
           2                        PRESENT_RECORD_INDEX,
           2                        (RECORD_NUMBER
           2                         -FIRST_RECORD+2),
           2                        (START_OF_ERROR1_REC
           2                         -FIRST_RECORD+2),
           2                        (END_OF_ERROR1_REC
           2                         -FIRST_RECORD+2))

            WRITE(3,77)ERROR1_GUESS,CHANNEL_COUNTER1,
```

```
     2               START_OF_ERROR1_REC,
     2               START_OF_ERROR1_SAMPLE,END_OF_ERROR1_REC,
     2               END_OF_ERROR1_SAMPLE
          CALL UPDATE_ERROR_STATISTICS(ERROR_STATISTICS,
     2                                 ERROR_LENGTH_STATISTICS,
     2                                 ERROR1_GUESS,
     2                                 START_OF_ERROR1_REC,
     2                                 END_OF_ERROR1_REC,
     2                                 START_OF_ERROR1_SAMPLE,
     2                                 END_OF_ERROR1_SAMPLE,
     2                                 CHANNEL_COUNTER1,
     2                                 TIME_BLOCK_COUNTER,
     2                                 TIME_SAMPLE_COUNTER)
          GO TO 250

C         AT 235 - NEVER USED, HERE JUST TO DUPLICATE
C                  THE BRANCH FOR THE CLEANUP OF THE
C                  SECOND ERROR
235       CONTINUE

          GO TO 250

C         AT 240 - UNKNOWN ERROR
240       CONTINUE

          CALL SET_ERROR_FLAG ((START_OF_ERROR1_REC
     2                          -FIRST_RECORD+2),
     2                         (END_OF_ERROR1_REC
     2                          -FIRST_RECORD+2),
     2                         START_OF_ERROR1_SAMPLE,
     2                         END_OF_ERROR1_SAMPLE,
     2                         CHANNEL_COUNTER1)
          CALL UPDATE_DATA_BLOCK (DATA_BLOCK,
     2                            PRESENT_RECORD_INDEX,
     2                            (RECORD_NUMBER
     2                             -FIRST_RECORD+2),
     2                            (START_OF_ERROR1_REC
     2                             -FIRST_RECORD+2),
     2                            (END_OF_ERROR1_REC
     2                             -FIRST_RECORD+2))
          WRITE(3,77)ERROR1_GUESS,CHANNEL_COUNTER1,
     2               START_OF_ERROR1_REC,
     2               START_OF_ERROR1_SAMPLE,END_OF_ERROR1_REC,
     2               END_OF_ERROR1_SAMPLE
          CALL UPDATE_ERROR_STATISTICS(ERROR_STATISTICS,
     2                                 ERROR_LENGTH_STATISTICS,
     2                                 ERROR1_GUESS,
     2                                 START_OF_ERROR1_REC,
     2                                 END_OF_ERROR1_REC,
     2                                 START_OF_ERROR1_SAMPLE,
     2                                 END_OF_ERROR1_SAMPLE,
     2                                 CHANNEL_COUNTER1,
     2                                 TIME_BLOCK_COUNTER,
     2                                 TIME_SAMPLE_COUNTER)
          CALL UPDATE_ERROR_STATISTICS(CLEAN_STATISTICS,
     2                                 CLEAN_LENGTH_STATISTICS,
     2                                 ERROR1_GUESS,
     2                                 START_OF_ERROR1_REC,
     2                                 END_OF_ERROR1_REC,
     2                                 START_OF_ERROR1_SAMPLE,
     2                                 END_OF_ERROR1_SAMPLE,
     2                                 CHANNEL_COUNTER1,
     2                                 TIME_BLOCK_COUNTER,
     2                                 TIME_SAMPLE_COUNTER)
```

```
            GO TO 250

C           AT 245 - NOT USED HERE, JUST A CODE DUPLICATION
245         CONTINUE

C           CLEAN UP THE SECOND ERROR

250         GO TO (255,260,265,270,275,280,285,290,295),
     2            CORRECTIVE_ACTION_CODE2

C           AT 255 - DO NOTHING
255         CONTINUE

            GO TO 300

C           AT 260 - CLEAR THE ERROR FLAG AND DO NOTHING ELSE
260         CONTINUE

            GO TO 300

C           AT 265 - SET VALUE TO CORRECTION VALUE
C                    AND CLEAR THE ERROR FLAG
265         CONTINUE

            GO TO 300

C           AT 270 - SET THE ERROR FLAG
270         CONTINUE

            GO TO 300

C           AT 275 - SET VALUE TO CORRECTION VALUE ONLY
C                    OVER SOME PART OF ERROR RUN
275         CONTINUE

            GO TO 300

C           AT 280 - RESCALE VALUE BY A MULTIPLE OF 2 TO
C                    GET AS CLOSE AS POSSIBLE TO CORRECTION VALUE
280         CONTINUE

            GO TO 300

C           AT 285 - CLEANING UP OF A STRING OF ERRORS
285         CONTINUE

            GO TO 202

C           AT 290 - UNKNOWN ERROR
290         CONTINUE

            GO TO 300

C           AT 295 - COMPUTE THE AVERAGE
295         CONTINUE


            CURRENT_CLEAN_VALUE=DATA_BLOCK(CHANNEL_COUNTER1,
     2                                     SAMPLE_COUNTER1,
     2                                     PRESENT_RECORD_INDEX)
     2                      .AND. DATA_MASK
            IF (SAMPLE_COUNTER1 .EQ. 1)THEN
               PREVIOUS_CLEAN_VALUE=DATA_BLOCK(CHANNEL_COUNTER1,
     2                                         320,
```

```
     2                                    PREVIOUS_INDEX
     2                                  (PRESENT_RECORD_INDEX))
     2                           .AND. DATA_MASK
          CALL TIME_CODE_CHECK
     2          (DATA_BLOCK(1,320,
     2                      PREVIOUS_INDEX
     2                       (PRESENT_RECORD_INDEX)),
     2           TIME_CODE_FLAG)
        ELSE
          PREVIOUS_CLEAN_VALUE=DATA_BLOCK(CHANNEL_COUNTER1,
     2                                    (SAMPLE_COUNTER1-1),
     2                                    PRESENT_RECORD_INDEX)
     2                           .AND. DATA_MASK
          CALL TIME_CODE_CHECK
     2          (DATA_BLOCK(1,(SAMPLE_COUNTER1-1),
     2                      PRESENT_RECORD_INDEX),
     2           TIME_CODE_FLAG)
        END IF
        IF (TIME_CODE_FLAG) THEN
          PREVIOUS_CLEAN_VALUE=AVERAGE(CHANNEL_COUNTER1)
          CALL SMOOTHNESS_TEST(2,
     2                         CURRENT_CLEAN_VALUE,
     2                         PREVIOUS_CLEAN_VALUE,
     2                         0,
     2                         0,
     2                         0,
     2                         SMOOTHNESS_CRITERIA
     2                           (CHANNEL_COUNTER1),
     2                         MAXIMUM_SMOOTH_JUMP
     2                           (CHANNEL_COUNTER1)*2,
     2                         CLEAN_SMOOTH_FLAG)
        ELSE
          CALL SMOOTHNESS_TEST(2,
     2                         CURRENT_CLEAN_VALUE,
     2                         PREVIOUS_CLEAN_VALUE,
     2                         0,
     2                         0,
     2                         0,
     2                         SMOOTHNESS_CRITERIA
     2                           (CHANNEL_COUNTER1),
     2                         MAXIMUM_SMOOTH_JUMP
     2                           (CHANNEL_COUNTER1),
     2                         CLEAN_SMOOTH_FLAG)
        END IF
        IF (.NOT. CLEAN_SMOOTH_FLAG) THEN
          WRITE (3,23)CHANNEL_COUNTER1,
     2                RECORD_NUMBER,
     2                SAMPLE_COUNTER1
          CORRECTIVE_ACTION_CODE1=START_UNMARKED_ERROR
          GO TO 202
        END IF

        AVERAGE(CHANNEL_COUNTER1)=(CURRENT_CLEAN_VALUE+
     2                             100.0*AVERAGE
     2                                  (CHANNEL_COUNTER1))
     2                            /101.0

C       ------------------------------------------------

300       CONTINUE
        END DO
350     CONTINUE
      END DO
```

```
C       ------------------------------------------------------

C       ------------------------------------------------------

400     RECORD_NUMBER=RECORD_NUMBER+1

        PRESENT_RECORD_INDEX=NEXT_INDEX(PRESENT_RECORD_INDEX)

        IF (RECORD_NUMBER .GT. LAST_RECORD)THEN
           GO TO 900
        END IF

        READ (1,REC=RECORD_NUMBER+1,ERR=3000)
        2  ((DATA_BLOCK(CHANNEL_COUNTER2,SAMPLE_COUNTER2,
        2               NEXT_INDEX(PRESENT_RECORD_INDEX)),
        2    CHANNEL_COUNTER2=1,8,1),
        2    SAMPLE_COUNTER2=1,320,1)

        WRITE(2,REC=RECORD_NUMBER+3-FIRST_RECORD,ERR=4000)
        2  ((DATA_BLOCK(CHANNEL_COUNTER2,SAMPLE_COUNTER2,
        2               NEXT_INDEX(PRESENT_RECORD_INDEX)),
        2    CHANNEL_COUNTER2=1,8,1),
        2    SAMPLE_COUNTER2=1,320,1)

        GOTO 100

C       END THE PROGRAM WHEN FINISHED PROCESSING
900     CONTINUE

        DO CLEANUP_COUNTER1 = 1,8,1
           IF (DATA_IN_CHANNEL(CLEANUP_COUNTER1) .NE. NO_DATA)THEN
              DATA_BLOCK(CLEANUP_COUNTER1,1,PRESENT_RECORD_INDEX)=0
              CORRECTIVE_ACTION_CODE1=FLUSH_OUT
              START_OF_ERROR1_SAMPLE=0
              CALL FOR_EACH_CHANNEL (CLEANUP_COUNTER1,
        2                            RECORD_NUMBER,
        2                            1,
        2                            PRESENT_RECORD_INDEX,
        2                            FIRST_RECORD,
        2                            DATA_BLOCK,
        2                            AVERAGE(CHANNEL_COUNTER1),
        2                            START_OF_ERROR1_REC,
        2                            START_OF_ERROR1_SAMPLE,
        2                            END_OF_ERROR1_REC,
        2                            END_OF_ERROR1_SAMPLE,
        2                            START_OF_ERROR2_REC,
        2                            START_OF_ERROR2_SAMPLE,
        2                            END_OF_ERROR2_REC,
        2                            END_OF_ERROR2_SAMPLE,
        2                            ERROR1_GUESS,
        2                            ERROR2_GUESS,
        2                            CORRECTIVE_ACTION_CODE1,
        2                            CORRECTION_VALUE1,
        2                            CORRECTIVE_ACTION_CODE2,
        2                            CORRECTION_VALUE2)
              IF (START_OF_ERROR1_SAMPLE .NE. 0)THEN
                 WRITE(3,16)CLEANUP_COUNTER1,START_OF_ERROR1_REC,
        2                   START_OF_ERROR1_SAMPLE
              END IF
           END IF
        END DO

        CLOSE(UNIT=1)
        CLOSE(UNIT=2)
```

```
DO PRINT_OUT_COUNTER3=1,TIME_BLOCK_COUNTER,1
   IF (PRINT_OUT_COUNTER3 .EQ. TIME_BLOCK_COUNTER)THEN
      ADJUSTED_TIME_BLOCK_LENGTH=TIME_SAMPLE_COUNTER/100.0
   ELSE
      ADJUSTED_TIME_BLOCK_LENGTH=900.0
   END IF
   WRITE(3,17)PRINT_OUT_COUNTER3
   DO PRINT_OUT_COUNTER1=1,17,1
      DO PRINT_OUT_COUNTER2=1,8,1
         IF(ERROR_STATISTICS(PRINT_OUT_COUNTER2,
2                            PRINT_OUT_COUNTER1,
2                            PRINT_OUT_COUNTER3) .NE. 0)THEN
            WRITE(3,19)
            WRITE(3,11)PRINT_OUT_COUNTER1,PRINT_OUT_COUNTER2,
2                    ERROR_STATISTICS(PRINT_OUT_COUNTER2,
2                                     PRINT_OUT_COUNTER1,
2                                     PRINT_OUT_COUNTER3),
2                ERROR_LENGTH_STATISTICS(PRINT_OUT_COUNTER2,
2                                     PRINT_OUT_COUNTER1,
2                                     PRINT_OUT_COUNTER3),
2                (ERROR_LENGTH_STATISTICS(PRINT_OUT_COUNTER2,
2                                      PRINT_OUT_COUNTER1,
2                                      PRINT_OUT_COUNTER3)/
2                 ADJUSTED_TIME_BLOCK_LENGTH)

            WRITE(3,21)
            WRITE(3,18)PRINT_OUT_COUNTER1,PRINT_OUT_COUNTER2,
2                    CLEAN_STATISTICS(PRINT_OUT_COUNTER2,
2                                     PRINT_OUT_COUNTER1,
2                                     PRINT_OUT_COUNTER3),
2                CLEAN_LENGTH_STATISTICS(PRINT_OUT_COUNTER2,
2                                     PRINT_OUT_COUNTER1,
2                                     PRINT_OUT_COUNTER3),
2               (CLEAN_LENGTH_STATISTICS(PRINT_OUT_COUNTER2,
2                                      PRINT_OUT_COUNTER1,
2                                      PRINT_OUT_COUNTER3)/
2                ADJUSTED_TIME_BLOCK_LENGTH)

         END IF
      END DO
   END DO
   DO LOCAL_COUNTER50=1,8,1
         COMBINED_BLOCK_ERRORS(LOCAL_COUNTER50)=0
         COMBINED_BLOCK_LENGTH(LOCAL_COUNTER50)=0
         COMBINED_CLEAN_ERRORS(LOCAL_COUNTER50)=0
         COMBINED_CLEAN_LENGTH(LOCAL_COUNTER50)=0
   END DO
   DO LOCAL_COUNTER50=1,8,1
      DO LOCAL_COUNTER51=1,17,1
         COMBINED_BLOCK_ERRORS(LOCAL_COUNTER50)=
2        COMBINED_BLOCK_ERRORS(LOCAL_COUNTER50)+
2        ERROR_STATISTICS(LOCAL_COUNTER50,LOCAL_COUNTER51,
2                         PRINT_OUT_COUNTER3)
         COMBINED_BLOCK_LENGTH(LOCAL_COUNTER50)=
2        COMBINED_BLOCK_LENGTH(LOCAL_COUNTER50)+
2        ERROR_LENGTH_STATISTICS(LOCAL_COUNTER50,
2                                LOCAL_COUNTER51,
2                                PRINT_OUT_COUNTER3)
         COMBINED_CLEAN_ERRORS(LOCAL_COUNTER50)=
2        COMBINED_CLEAN_ERRORS(LOCAL_COUNTER50)+
2        CLEAN_STATISTICS(LOCAL_COUNTER50,LOCAL_COUNTER51,
2                         PRINT_OUT_COUNTER3)
         COMBINED_CLEAN_LENGTH(LOCAL_COUNTER50)=
2        COMBINED_CLEAN_LENGTH(LOCAL_COUNTER50)+
```

```
2        CLEAN_LENGTH_STATISTICS(LOCAL_COUNTER50,
2                                LOCAL_COUNTER51,
2                                PRINT_OUT_COUNTER3)
      END DO
   END DO
   DO LOCAL_COUNTER50=1,8,1
      WRITE(3,19)
      WRITE(3,24)LOCAL_COUNTER50,
2               COMBINED_BLOCK_ERRORS(LOCAL_COUNTER50),
2               COMBINED_BLOCK_LENGTH(LOCAL_COUNTER50)
      WRITE(3,21)
      WRITE(3,24)LOCAL_COUNTER50,
2               COMBINED_CLEAN_ERRORS(LOCAL_COUNTER50),
2               COMBINED_CLEAN_LENGTH(LOCAL_COUNTER50)
   END DO
END DO


DO SUMMARY_COUNTER1=1,TIME_BLOCK_COUNTER,1
   DO SUMMARY_COUNTER2=1,8,1
      DO SUMMARY_COUNTER3=1,17,1
         SUMMARY_STATISTICS(SUMMARY_COUNTER2,
2                           SUMMARY_COUNTER3)=
2        SUMMARY_STATISTICS(SUMMARY_COUNTER2,
2                           SUMMARY_COUNTER3)+
2        ERROR_STATISTICS(SUMMARY_COUNTER2,
2                         SUMMARY_COUNTER3,
2                         SUMMARY_COUNTER1)
         CLEAN_SUMMARY_STATISTICS(SUMMARY_COUNTER2,
2                                 SUMMARY_COUNTER3)=
2        CLEAN_SUMMARY_STATISTICS(SUMMARY_COUNTER2,
2                                 SUMMARY_COUNTER3)+
2        CLEAN_STATISTICS(SUMMARY_COUNTER2,
2                         SUMMARY_COUNTER3,
2                         SUMMARY_COUNTER1)
         SUMMARY_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                                  SUMMARY_COUNTER3)=
2        SUMMARY_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                                  SUMMARY_COUNTER3)+
2        ERROR_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                                SUMMARY_COUNTER3,
2                                SUMMARY_COUNTER1)
         CLEAN_SUMMARY_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                                        SUMMARY_COUNTER3)=
2        CLEAN_SUMMARY_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                                        SUMMARY_COUNTER3)+
2        CLEAN_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                                SUMMARY_COUNTER3,
2                                SUMMARY_COUNTER1)

      END DO
   END DO
END DO

ADJUSTED_TOTAL_LENGTH=(LAST_RECORD+1-FIRST_RECORD)*3.2

DO SUMMARY_COUNTER1=1,17,1
   DO SUMMARY_COUNTER2=1,8,1
      SUMMARY_PERCENTAGE (SUMMARY_COUNTER2,
2                         SUMMARY_COUNTER1)=
2     SUMMARY_LENGTH_STATISTICS(SUMMARY_COUNTER2,
2                               SUMMARY_COUNTER1)/
2     ADJUSTED_TOTAL_LENGTH
      SUMMARY_CLEAN_PERCENTAGE (SUMMARY_COUNTER2,
```

```
     2                              SUMMARY_COUNTER1)=
     2     CLEAN_SUMMARY_LENGTH_STATISTICS(SUMMARY_COUNTER2,
     2                                    SUMMARY_COUNTER1)/
     2     ADJUSTED_TOTAL_LENGTH
        END DO
     END DO


     WRITE(3,22)
     DO PRINT_OUT_COUNTER1=1,17,1
        DO PRINT_OUT_COUNTER2=1,8,1
           IF (DATA_IN_CHANNEL(PRINT_OUT_COUNTER2)
     2         .NE. NO_DATA)THEN
             WRITE(3,19)
             WRITE(3,11)PRINT_OUT_COUNTER1,PRINT_OUT_COUNTER2,
     2                SUMMARY_STATISTICS(PRINT_OUT_COUNTER2,
     2                                   PRINT_OUT_COUNTER1),
     2            SUMMARY_LENGTH_STATISTICS(PRINT_OUT_COUNTER2,
     2                                   PRINT_OUT_COUNTER1),
     2                 SUMMARY_PERCENTAGE (PRINT_OUT_COUNTER2,
     2                                   PRINT_OUT_COUNTER1)

             WRITE(3,21)
             WRITE(3,18)PRINT_OUT_COUNTER1,PRINT_OUT_COUNTER2,
     2             CLEAN_SUMMARY_STATISTICS(PRINT_OUT_COUNTER2,
     2                                   PRINT_OUT_COUNTER1),
     2      CLEAN_SUMMARY_LENGTH_STATISTICS(PRINT_OUT_COUNTER2,
     2                                   PRINT_OUT_COUNTER1),
     2            SUMMARY_CLEAN_PERCENTAGE (PRINT_OUT_COUNTER2,
     2                                   PRINT_OUT_COUNTER1)
           END IF
        END DO
     END DO
     CLOSE(UNIT=3)
     STOP


C    END THE PROGRAM WHEN THERE WERE NOT 3 SPECIFIED RECORDS IN THE
C    INPUT FILE
1000 CONTINUE
     CLOSE(UNIT=1)
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     STOP

C    END THE PROGRAM WHEN THERE WAS NOT SPACE FOR 3 INITIAL RECORDS
C    IN THE OUTPUT FILE.
2000 CONTINUE
     CLOSE(UNIT=1)
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     STOP

C    END THE PROGRAM WHEN THERE ARE NO MORE INPUT RECORDS
3000 CONTINUE
     CLOSE(UNIT=1)
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     STOP

C    END THE PROGRAM WHEN THERE IS NO MORE SPACE IN THE OUTPUT FILE.
4000 CONTINUE
     CLOSE(UNIT=1)
     CLOSE(UNIT=2)
```

```
CLOSE(UNIT=3)
STOP
END
```

E.2 Error_type Subroutine

```
      SUBROUTINE ERROR_TYPE (CHANNEL_NUMBER,
     2                       RECORD_NUMBER,
     2                       SAMPLE_COUNTER1,
     2                       PRESENT_RECORD_INDEX,
     2                       FIRST_RECORD,
     2                       DATA_BLOCK,
     2                       AVERAGE_VALUE,
     2                       START_OF_ERROR1_REC,
     2                       START_OF_ERROR1_SAMPLE,
     2                       END_OF_ERROR1_REC,
     2                       END_OF_ERROR1_SAMPLE,
     2                       START_OF_ERROR2_REC,
     2                       START_OF_ERROR2_SAMPLE,
     2                       END_OF_ERROR2_REC,
     2                       END_OF_ERROR2_SAMPLE,
     2                       ERROR1_GUESS,
     2                       ERROR2_GUESS,
     2                       CORRECTIVE_ACTION_CODE1,
     2                       CORRECTION_VALUE1,
     2                       CORRECTIVE_ACTION_CODE2,
     2                       CORRECTION_VALUE2)

C     PARAMETER DECLARATIONS

      INTEGER*2 SMOOTH_CONTINUOUS,
     2          UBNORMALLY_CONSTANT,
     2          CONSTANT_LAST_VALUE,
     2          CONSTANT_LAST_TO_1023,
     2          CONSTANT,
     2          CONSTANT_AT_1023,
     2          CONSTANT_PLUS_ENDS,
     2          SEVERAL_CONST_PLUS_ENDS,
     2          SEVERAL_SHIFTED,
     2          UNKNOWN_VARIABILITY,
     2          VARIABILITY_TYPES,
     2          MARKED,
     2          UNMARKED,
     2          STARTS_AS_UNMARKED,
     2          MARKED_TYPES,
     2          ENTERED_SMOOTHLY,
     2          ENTERED_ABRUPTLY,
     2          ENTRANCE_UNKNOWN,
     2          ENTRANCE_TYPES,
     2          EXITED_SMOOTHLY,
     2          EXITED_ABRUPTLY,
     2          EXIT_UNKNOWN,
     2          EXIT_TYPES,
     2          ZERO_LENGTH,
     2          LESS_THAN_21,
     2          BETWEEN_21_AND_100,
     2          BETWEEN_101_AND_200,
     2          MORE_THAN_200,
     2          ERROR_LENGTH_TYPES,
     2          UNMARKED_LENGTH_TYPES,
     2          NO_ERROR,
     2          G_1_1,
     2          G_1_2,
     2          G_1_3,
     2          G_1_4,
     2          G_2_1,
     2          G_2_2,
     2          G_2_3,
```

```
     2         G_3_1,
     2         G_3_2,
     2         G_3_3,
     2         G_3_4,
     2         G_3_5,
     2         G_4_1,
     2         G_4_2,
     2         G_4_3,
     2         G_4_4,
     2         UNKNOWN_ERROR,
     2         DO_NOTHING,
     2         CLEAR_FLAG,
     2         SET_FLAG,
     2         CORRECT_AND_CLEAR,
     2         REPORT,
     2         RESCALE,
     2         EX_UNMARKED,
     2         UNKNOWN_ERROR_ACTION,
     2         FLUSH_OUT,
     2         START_UNMARKED_ERROR,
     2         CERTAIN_END,
     2         UNSURE,
     2         NOT_AN_END,
     2         NUMBER_OF_ERROR_VALUES,
     2         ROUGH_VALUE_PARAMETER

     INTEGER*2 DATA_MASK,
     2         ERROR_MASK

     INTEGER*4 MAXIMUM_LENGTH,
     2         MAXIMUM_UNMARKED_LENGTH

C    VARIABILITY CODES:

     PARAMETER (SMOOTH_CONTINUOUS = 1)
     PARAMETER (UBNORMALLY_CONSTANT = 2)
     PARAMETER (CONSTANT_LAST_VALUE = 3)
     PARAMETER (CONSTANT_LAST_TO_1023 = 4)
     PARAMETER (CONSTANT = 5)
     PARAMETER (CONSTANT_AT_1023 = 6)
     PARAMETER (CONSTANT_PLUS_ENDS = 7)
     PARAMETER (SEVERAL_CONST_PLUS_ENDS = 8)
     PARAMETER (SEVERAL_SHIFTED = 9)
     PARAMETER (UNKNOWN_VARIABILITY = 10)
     PARAMETER (VARIABILITY_TYPES = 10)

C    MARKED TYPES CODES:

     PARAMETER (MARKED = 1)
     PARAMETER (UNMARKED = 2)
     PARAMETER (STARTS_AS_UNMARKED = 3)
     PARAMETER (MARKED_TYPES = 3)

C    ENTRANCE TYPES CODES:

     PARAMETER (ENTERED_SMOOTHLY = 1)
     PARAMETER (ENTERED_ABRUPTLY = 2)
     PARAMETER (ENTRANCE_UNKNOWN = 3)
     PARAMETER (ENTRANCE_TYPES = 3)

C    EXIT TYPES CODES:

     PARAMETER (EXITED_SMOOTHLY  = 1)
     PARAMETER (EXITED_ABRUPTLY  = 2)
```

```
      PARAMETER (EXIT_UNKNOWN      = 3)
      PARAMETER (EXIT_TYPES = 3)

C     LENGTH TYPES CODES:

      PARAMETER (ZERO_LENGTH    = 1)
      PARAMETER (LESS_THAN_21   = 2)
      PARAMETER (BETWEEN_21_AND_100 = 3)
      PARAMETER (BETWEEN_101_AND_200 = 4)
      PARAMETER (MORE_THAN_200 = 5)
      PARAMETER (ERROR_LENGTH_TYPES = 5)

C     ERROR CODES:
      PARAMETER (NO_ERROR = 0)
      PARAMETER (G_1_1 = 1)
      PARAMETER (G_1_2 = 2)
      PARAMETER (G_1_3 = 3)
      PARAMETER (G_1_4 = 4)
      PARAMETER (G_2_1 = 5)
      PARAMETER (G_2_2 = 6)
      PARAMETER (G_2_3 = 7)
      PARAMETER (G_3_1 = 8)
      PARAMETER (G_3_2 = 9)
      PARAMETER (G_3_3 = 10)
      PARAMETER (G_3_4 = 11)
      PARAMETER (G_3_5 = 12)
      PARAMETER (G_4_1 = 13)
      PARAMETER (G_4_2 = 14)
      PARAMETER (G_4_3 = 15)
      PARAMETER (G_4_4 = 16)
      PARAMETER (UNKNOWN_ERROR = 17)

C     CORRECTIVE ACTION CODES:
      PARAMETER (DO_NOTHING = 1)
      PARAMETER (CLEAR_FLAG = 2)
      PARAMETER (SET_FLAG =4)
      PARAMETER (CORRECT_AND_CLEAR = 3)
      PARAMETER (REPORT = 5)
      PARAMETER (RESCALE = 6)
      PARAMETER (EX_UNMARKED = 7)
      PARAMETER (UNKNOWN_ERROR_ACTION = 8)
      PARAMETER (FLUSH_OUT = 10)
      PARAMETER (START_UNMARKED_ERROR =11)

      PARAMETER (NUMBER_OF_ERROR_VALUES = 100)

      PARAMETER (DATA_MASK = 1023)
      PARAMETER (ERROR_MASK = 2048)

C     END OF ERROR DESCRIPTIONS:
      PARAMETER (CERTAIN_END = 0)
      PARAMETER (UNSURE = 1)
      PARAMETER (NOT_AN_END = 2)

      PARAMETER (MAXIMUM_LENGTH = 200)

      PARAMETER (ROUGH_VALUE_PARAMETER = 5)

      PARAMETER (MAXIMUM_UNMARKED_LENGTH = 320)


      INTEGER*2 ERROR_ID_ARRAY (VARIABILITY_TYPES,
     2                          ENTRANCE_TYPES,
     2                          EXIT_TYPES,
```

```
2                       ERROR_LENGTH_TYPES,
2                       MARKED_TYPES),
2         CHANNEL_NUMBER,
2         ERROR1_GUESS,
2         ERROR2_GUESS,
2         CORRECTIVE_ACTION_CODE1,
2         CORRECTIVE_ACTION_CODE2,
2         ERROR_END(8),
2         VARIABILITY,
2         ENTRANCE,
2         EXIT,
2         ERROR_LENGTH_CATEGORY,
2         UNMARKED_CATEGORY,
2         LOCAL_COUNTER1,
2         PRESENT_RECORD_INDEX,
2         NEXT_INDEX(3)/2,3,1/,
2         PREVIOUS_INDEX(3)/3,1,2/,
2         POINTS_IN_ENTRANCE_BLOCK /0/,
2         POINTS_IN_EXIT_BLOCK /0/,
2         SMOOTHNESS_TEST_INDEX /1/

INTEGER*2 MARKED_ERROR (8),
2         SAMPLE_COUNTER1,
2         START_OF_ERROR1_SAMPLE,
2         START_OF_ERROR2_SAMPLE,
2         END_OF_ERROR1_SAMPLE,
2         END_OF_ERROR2_SAMPLE,
2         POSSIBLE_ERROR_START_SAMPLE(8,1000),
2         POSSIBLE_ERROR_END_SAMPLE(8,1000),
2         ENTRANCE_BLOCK_START_SAMPLE /0/,
2         EXIT_BLOCK_END_SAMPLE /0/,
2         START_UNSMOOTH /0/,
2         END_UNSMOOTH /0/,
2         CORRECTION_VALUE1,
2         CORRECTION_VALUE2,
2         DATA_VALUE /0/,
2         CLEAN_DATA /0/,
2         DATA_BLOCK(8,320,3),
2         SCRATCH_BLOCK(8,320) /2560*0/,
2         ERROR_BLOCK(320) /320*0/,
2         CLEAN_ERROR_BLOCK(-4:325) /330*0/,
2         ENTRANCE_BLOCK(5) /5*0/,
2         EXIT_BLOCK(5) /5*0/,
2         VALUES_OF_ERROR(NUMBER_OF_ERROR_VALUES),
2         UNMARKED_PORTION_VALUES(8,5) /40*0/,
2         ROUGH_VALUE_DISTANCE(8)/8*0/,
2         CURRENT_POSSIBLE_ERROR(8) /8*1/,
2         ERROR_IN_PROGRESS /1/,
2         VARIABILITY_COUNTER /1/,
2         ERROR_COUNTER1 /1/,
2         COUNTER2 /1/,
2         COUNTER3 /1/,
2         LOCAL_COUNTER2 /1/,
2         LOCAL_COUNTER3 /1/,
2         LOCAL_COUNTER4 /1/,
2         LOCAL_COUNTER5 /1/,
2         LOCAL_COUNTER6 /1/,
2         LOCAL_COUNTER7 /1/,
2         LOCAL_COUNTER8 /1/,
2         LOCAL_COUNTER20 /0/,
2         LOCAL_COUNTER25 /1/,
2         LOCAL_COUNTER26 /1/,
2         LOCAL_COUNTER27 /1/,
2         LOCAL_COUNTER28 /1/,
```

```
     2         LOCAL_COUNTER29 /1/,
     2         LOCAL_COUNTER30 /1/,
     2         SAMPLE_COUNTER3 /1/,
     2         SAMPLE_COUNTER4 /1/,
     2         SAMPLE_COUNTER5 /1/,
     2         CHANNEL_COUNTER3 /1/,
     2         CHANNEL_COUNTER4 /1/,
     2         CHANNEL_COUNTER5 /1/,
     2         ERROR_BIT,
     2         LONG_CONSTANT_VALUE(8)/8*0/,
     2         MAXIMUM_SMOOTH_JUMP(8),
     2         LIGHT_LIMIT(8)/4*20,60,50,2*60/,
     2         TIGHT_LIMIT(8)/4*10,20,25,2*20/,
     2         ORIGINAL_ERROR_BLOCK_VALUE /0/,
     2         EIGHT_CHANNELS_VALUES(8)/8*0/,
     2         NUMBER_OF_TIME_CODES /0/,
     2         TIME_SHIFT_INDEX /0/

     INTEGER*4 ERROR_LENGTH_COUNTER (8)/8*0/,
     2         UNMARKED_PORTION_COUNTER (8),
     2         START_OF_ERROR1_REC,
     2         START_OF_ERROR2_REC,
     2         END_OF_ERROR1_REC,
     2         END_OF_ERROR2_REC,
     2         POSSIBLE_ERROR_START_REC(8,1000)/8000*0/,
     2         POSSIBLE_ERROR_END_REC(8,1000)/8000*0/,
     2         ENTRANCE_BLOCK_START_REC /0/,
     2         EXIT_BLOCK_END_REC /0/,
     2         RECORD_NUMBER,
     2         FIRST_RECORD,
     2         ERROR_LENGTH /0/

     REAL*4 SMOOTHNESS_CRITERIA(8),
     2      AVERAGE_VALUE

     LOGICAL*1 LONG_CONSTANT(8)/8*.FALSE./,
     2         NOT_A_LONG_CONSTANT(8)/8*.FALSE./,
     2         UNMARKED_PORTION_SMOOTH,
     2         SWITCHED_TO_1023(8),
     2         SMOOTHNESS_FLAG /.TRUE./,
     2         ALREADY_SHIFTED /.FALSE./,
     2         BEGINNING_OF_ERROR_SMOOTH /.TRUE./,
     2         END_OF_ERROR_SMOOTH /.TRUE./,
     2         ENTIRELY_CONSTANT /.TRUE./,
     2         TIME_CODE_FLAG /.FALSE./,
     2         TIME_CODE_IN_ERROR(MAXIMUM_LENGTH)
     2           /MAXIMUM_LENGTH * .FALSE./,
     2         INITIALIZE_ID_ARRAY /.TRUE./,
     2         POSSIBLE_SHIFTED_ERROR /.TRUE./,
     2         LONG_CONSTANT_ANALYSED(8) /8 * .FALSE./


     COMMON /BLOCK1/ SMOOTHNESS_CRITERIA,
     2               MAXIMUM_SMOOTH_JUMP

     IF (INITIALIZE_ID_ARRAY)THEN
C       INITIALIZE ERROR_ID_ARRAY. DONE ONLY ONCE - THE FIRST TIME
C       THIS SUBROUTINE IS CALLED
        DO LOCAL_COUNTER25 =1,VARIABILITY_TYPES,1
           DO LOCAL_COUNTER26 = 1,ENTRANCE_TYPES,1
              DO LOCAL_COUNTER27 =1,EXIT_TYPES,1
                 DO LOCAL_COUNTER28 =1,ERROR_LENGTH_TYPES,1
                    DO LOCAL_COUNTER29=1,MARKED_TYPES,1
           ERROR_ID_ARRAY(LOCAL_COUNTER25,
```

```
     2                    LOCAL_COUNTER26,
     2                    LOCAL_COUNTER27,
     2                    LOCAL_COUNTER28,
     2                    LOCAL_COUNTER29)=UNKNOWN_ERROR
                       END DO
                    END DO
                 END DO
              END DO
           END DO
C          PUT CODES FOR ALL OTHER ERRORS HERE:

           ERROR_ID_ARRAY(SMOOTH_CONTINUOUS,
     2                    ENTERED_SMOOTHLY,
     2                    EXITED_SMOOTHLY,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_1

           ERROR_ID_ARRAY(SMOOTH_CONTINUOUS,
     2                    ENTRANCE_UNKNOWN,
     2                    EXITED_SMOOTHLY,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_1

           ERROR_ID_ARRAY(SMOOTH_CONTINUOUS,
     2                    ENTERED_SMOOTHLY,
     2                    EXIT_UNKNOWN,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_1

           ERROR_ID_ARRAY(CONSTANT_LAST_VALUE,
     2                    ENTERED_SMOOTHLY,
     2                    EXITED_SMOOTHLY,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_2

           ERROR_ID_ARRAY(CONSTANT_LAST_VALUE,
     2                    ENTRANCE_UNKNOWN,
     2                    EXITED_SMOOTHLY,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_2

           ERROR_ID_ARRAY(CONSTANT_LAST_VALUE,
     2                    ENTERED_SMOOTHLY,
     2                    EXIT_UNKNOWN,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_2

           ERROR_ID_ARRAY(CONSTANT,
     2                    ENTERED_SMOOTHLY,
     2                    EXITED_SMOOTHLY,
     2                    LESS_THAN_21,
     2                    MARKED)=
     2                    G_1_3

           ERROR_ID_ARRAY(CONSTANT,
     2                    ENTRANCE_UNKNOWN,
     2                    EXITED_SMOOTHLY,
     2                    LESS_THAN_21,
```

```
     2                MARKED)=
     2                G_1_3

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_1_3

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_1_4

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_1_4

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_1_4

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_1_4

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_1_4

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_1_4

C       ------------------------------------------------------

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                MARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_ABRUPTLY,
     2                EXITED_SMOOTHLY,
```

```
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTRANCE_UNKNOWN,
2                EXITED_SMOOTHLY,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_SMOOTHLY,
2                EXITED_ABRUPTLY,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_ABRUPTLY,
2                EXITED_ABRUPTLY,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTRANCE_UNKNOWN,
2                EXITED_ABRUPTLY,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_SMOOTHLY,
2                EXIT_UNKNOWN,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_ABRUPTLY,
2                EXIT_UNKNOWN,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTRANCE_UNKNOWN,
2                EXIT_UNKNOWN,
2                MORE_THAN_200,
2                MARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_SMOOTHLY,
2                EXITED_SMOOTHLY,
2                MORE_THAN_200,
2                UNMARKED)=
2                G_2_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_ABRUPTLY,
2                EXITED_SMOOTHLY,
2                MORE_THAN_200,
```

```
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_ABRUPTLY,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_ABRUPTLY,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_1

C       ------------------------------------------------------
        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                MARKED)=
     2                G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                ENTERED_ABRUPTLY,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
```

```
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 MORE_THAN_200,
2                 UNMARKED)=
2                 G_2_2

   ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
2                 MORE_THAN_200,
2                 UNMARKED)=
```

```
     2                G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTRANCE_UNKNOWN,
     2                 EXITED_SMOOTHLY,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTERED_SMOOTHLY,
     2                 EXITED_ABRUPTLY,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTERED_ABRUPTLY,
     2                 EXITED_ABRUPTLY,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTRANCE_UNKNOWN,
     2                 EXITED_ABRUPTLY,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTERED_SMOOTHLY,
     2                 EXIT_UNKNOWN,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTERED_ABRUPTLY,
     2                 EXIT_UNKNOWN,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

        ERROR_ID_ARRAY(CONSTANT_LAST_TO_1023,
     2                 ENTRANCE_UNKNOWN,
     2                 EXIT_UNKNOWN,
     2                 MORE_THAN_200,
     2                 UNMARKED)=
     2                 G_2_2

C    ------------------------------------------------

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                 ENTERED_SMOOTHLY,
     2                 EXITED_SMOOTHLY,
     2                 MORE_THAN_200,
     2                 MARKED)=
     2                 G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                 ENTERED_ABRUPTLY,
     2                 EXITED_SMOOTHLY,
```

```
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 MORE_THAN_200,
2                 MARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 MORE_THAN_200,
2                 UNMARKED)=
2                 G_2_3

   ERROR_ID_ARRAY(CONSTANT_AT_1023,
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
2                 MORE_THAN_200,
```

```
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTERED_SMOOTHLY,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTERED_ABRUPTLY,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTERED_ABRUPTLY,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3

        ERROR_ID_ARRAY(CONSTANT_AT_1023,
     2                ENTRANCE_UNKNOWN,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                UNMARKED)=
     2                G_2_3
C    ------------------------------------------------------
        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                UNMARKED)=
```

```
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                LESS_THAN_21,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_21_AND_100,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                UNMARKED)=
     2                G_3_1

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                UNMARKED)=
     2                G_3_1

C       ------------------------------------------------

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
```

```
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_2

        ERROR_ID_ARRAY(SEVERAL_CONST_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_2

C       ------------------------------------------------------
        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
```

```
     2                LESS_THAN_21,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                LESS_THAN_21,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_21_AND_100,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                UNMARKED)=
     2                G_3_3

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                UNMARKED)=
     2                G_3_3

C       ------------------------------------------------

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
```

```
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_4

        ERROR_ID_ARRAY(CONSTANT_PLUS_ENDS,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_4
C       ------------------------------------------------------
        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                UNMARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
```

```
2                 LESS_THAN_21,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_101_AND_200,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_101_AND_200,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 BETWEEN_101_AND_200,
2                 UNMARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 MARKED)=
2                 G_3_5

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
```

```
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                LESS_THAN_21,
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_21_AND_100,
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_5

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                MARKED)=
     2                G_3_5

C       ------------------------------------------------------

        DO LOCAL_COUNTER30=SMOOTH_CONTINUOUS,
     2                    UNKNOWN_VARIABILITY,1
        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
```

```
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_ABRUPTLY,
```

```
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_101_AND_200,
2                 STARTS_AS_UNMARKED)=
2                 G_4_3

   ERROR_ID_ARRAY(LOCAL_COUNTER30,
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
```

```
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_SMOOTHLY,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_ABRUPTLY,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_ABRUPTLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTRANCE_UNKNOWN,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_3

        END DO

C       ------------------------------------------------

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1
```

```
   ERROR_ID_ARRAY(CONSTANT,
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_1

   ERROR_ID_ARRAY(CONSTANT,
```

```
2                ENTERED_ABRUPTLY,
2                EXITED_SMOOTHLY,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTRANCE_UNKNOWN,
2                EXITED_SMOOTHLY,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_SMOOTHLY,
2                EXITED_ABRUPTLY,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_ABRUPTLY,
2                EXITED_ABRUPTLY,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTRANCE_UNKNOWN,
2                EXITED_ABRUPTLY,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_SMOOTHLY,
2                EXIT_UNKNOWN,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_ABRUPTLY,
2                EXIT_UNKNOWN,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTRANCE_UNKNOWN,
2                EXIT_UNKNOWN,
2                BETWEEN_21_AND_100,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_SMOOTHLY,
2                EXITED_SMOOTHLY,
2                BETWEEN_101_AND_200,
2                STARTS_AS_UNMARKED)=
2                G_4_1

   ERROR_ID_ARRAY(CONSTANT,
2                ENTERED_ABRUPTLY,
```

```
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_ABRUPTLY,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTERED_ABRUPTLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

        ERROR_ID_ARRAY(CONSTANT,
     2                ENTRANCE_UNKNOWN,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_1

C       ------------------------------------------------

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                LESS_THAN_21,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2
```

```
   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 LESS_THAN_21,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
```

```
2                 ENTERED_ABRUPTLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_ABRUPTLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_ABRUPTLY,
2                 EXITED_ABRUPTLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXITED_ABRUPTLY,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_ABRUPTLY,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTRANCE_UNKNOWN,
2                 EXIT_UNKNOWN,
2                 BETWEEN_21_AND_100,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_SMOOTHLY,
2                 EXITED_SMOOTHLY,
2                 BETWEEN_101_AND_200,
2                 STARTS_AS_UNMARKED)=
2                 G_4_2

   ERROR_ID_ARRAY(SEVERAL_SHIFTED,
2                 ENTERED_ABRUPTLY,
```

```
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_ABRUPTLY,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_ABRUPTLY,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTERED_ABRUPTLY,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

        ERROR_ID_ARRAY(SEVERAL_SHIFTED,
     2                ENTRANCE_UNKNOWN,
     2                EXIT_UNKNOWN,
     2                BETWEEN_101_AND_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_2

C       ------------------------------------------------------

        DO LOCAL_COUNTER30=SMOOTH_CONTINUOUS,
     2                    UNKNOWN_VARIABILITY,1
        ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_SMOOTHLY,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4
```

```
      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_ABRUPTLY,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_SMOOTHLY,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_SMOOTHLY,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_ABRUPTLY,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTRANCE_UNKNOWN,
     2                EXITED_ABRUPTLY,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_SMOOTHLY,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTERED_ABRUPTLY,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      ERROR_ID_ARRAY(LOCAL_COUNTER30,
     2                ENTRANCE_UNKNOWN,
     2                EXIT_UNKNOWN,
     2                MORE_THAN_200,
     2                STARTS_AS_UNMARKED)=
     2                G_4_4

      END DO
C     ------------------------------------------------

C     DONE INITIALIZING
      INITIALIZE_ID_ARRAY = .FALSE.
    END IF
```

```
      IF  (CORRECTIVE_ACTION_CODE1 .EQ. FLUSH_OUT)THEN
         START_OF_ERROR1_REC=POSSIBLE_ERROR_START_REC
     2                       (CHANNEL_NUMBER,1)
         START_OF_ERROR1_SAMPLE=POSSIBLE_ERROR_START_SAMPLE
     2                         (CHANNEL_NUMBER,1)
         RETURN
      END IF

      IF  (CORRECTIVE_ACTION_CODE2 .EQ. EX_UNMARKED) THEN
C        CLASSIFY AND RETURN NEXT PORTION + INFO
         GO TO 50
      END IF

      ERROR_LENGTH_COUNTER(CHANNEL_NUMBER)=
     2  ERROR_LENGTH_COUNTER(CHANNEL_NUMBER)+1

C     GET THE ACTUAL DATA VALUE

      DATA_VALUE=DATA_BLOCK(CHANNEL_NUMBER,SAMPLE_COUNTER1,
     2                      PRESENT_RECORD_INDEX)
      CLEAN_DATA = DATA_VALUE .AND. DATA_MASK

C     GET THE ERROR FLAG:

      ERROR_BIT = DATA_VALUE .AND. ERROR_MASK
      IF ((ERROR_BIT .NE. ERROR_MASK)
     2   .AND.
     2   (CORRECTIVE_ACTION_CODE1 .NE. START_UNMARKED_ERROR))
     2               THEN

C        DO THE FOLLOWING IF ERROR BIT IS CLEARED:

         UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER) =
     2  UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)+1

C        FIND IF THIS IS THE ERROR END:
         IF (UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)
     2      .LT. 5)THEN
           GO TO (30,31,32,33), UNMARKED_PORTION_COUNTER
     2                            (CHANNEL_NUMBER)
30         UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)=CLEAN_DATA
           CALL SMOOTHNESS_TEST (1,
     2                           UNMARKED_PORTION_VALUES
     2                             (CHANNEL_NUMBER,5),
     2                           0,
     2                           0,
     2                           0,
     2                           0,
     2                           SMOOTHNESS_CRITERIA
     2                             (CHANNEL_NUMBER),
     2                           MAXIMUM_SMOOTH_JUMP
     2                             (CHANNEL_NUMBER),
     2                           UNMARKED_PORTION_SMOOTH)
           GO TO 34
31         CONTINUE
           UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,4)=
     2     UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)
           UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)=
     2     CLEAN_DATA
           CALL SMOOTHNESS_TEST (2,
     2                           UNMARKED_PORTION_VALUES
     2                             (CHANNEL_NUMBER,5),
     2                           UNMARKED_PORTION_VALUES
```

```
     2                          (CHANNEL_NUMBER,4),
     2                         0,
     2                         0,
     2                         0,
     2                         SMOOTHNESS_CRITERIA
     2                          (CHANNEL_NUMBER),
     2                         MAXIMUM_SMOOTH_JUMP
     2                          (CHANNEL_NUMBER),
     2                         UNMARKED_PORTION_SMOOTH)
          GO TO 34
32        CONTINUE
          UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,3)=
     2    UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,4)
          UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,4)=
     2    UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)
          UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)=CLEAN_DATA
          CALL SMOOTHNESS_TEST (3,
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,5),
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,4),
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,3),
     2                         0,
     2                         0,
     2                         SMOOTHNESS_CRITERIA
     2                          (CHANNEL_NUMBER),
     2                         MAXIMUM_SMOOTH_JUMP
     2                          (CHANNEL_NUMBER),
     2                         UNMARKED_PORTION_SMOOTH)
          GO TO 34
33        CONTINUE
          DO LOCAL_COUNTER1=2,4,1
             UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,
     2                               LOCAL_COUNTER1)=
     2       UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,
     2                               (LOCAL_COUNTER1+1))
          END DO
          UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)=CLEAN_DATA
          CALL SMOOTHNESS_TEST (4,
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,5),
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,4),
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,3),
     2                         UNMARKED_PORTION_VALUES
     2                          (CHANNEL_NUMBER,2),
     2                         0,
     2                         SMOOTHNESS_CRITERIA
     2                          (CHANNEL_NUMBER),
     2                         MAXIMUM_SMOOTH_JUMP
     2                          (CHANNEL_NUMBER),
     2                         UNMARKED_PORTION_SMOOTH)

34        CONTINUE
          IF(UNMARKED_PORTION_SMOOTH) THEN
             IF(ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER) .GT. 0)THEN
                ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)=
     2          ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)-1
                ERROR_END(CHANNEL_NUMBER)=NOT_AN_END
             ELSE
                IF (ABS(CLEAN_DATA-AVERAGE_VALUE)
     2              .LE. LIGHT_LIMIT(CHANNEL_NUMBER)) THEN
```

```
            ERROR_END(CHANNEL_NUMBER)=UNSURE
          ELSE
            ERROR_END(CHANNEL_NUMBER)=NOT_AN_END
          END IF
        END IF
      ELSE
        UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)=0
        ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)=ROUGH_VALUE_PARAMETER
        ERROR_END(CHANNEL_NUMBER)=NOT_AN_END
      END IF
    ELSE !-ELSE FROM "UNMARKED_PORTION_COUNTER .LT.5 IF"
      DO LOCAL_COUNTER1=1,4,1
        UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,
2                               LOCAL_COUNTER1)=
2       UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,
2                               (LOCAL_COUNTER1+1))
      END DO
      UNMARKED_PORTION_VALUES(CHANNEL_NUMBER,5)=CLEAN_DATA
      CALL SMOOTHNESS_TEST (5,
2                           UNMARKED_PORTION_VALUES
2                             (CHANNEL_NUMBER,5),
2                           UNMARKED_PORTION_VALUES
2                             (CHANNEL_NUMBER,4),
2                           UNMARKED_PORTION_VALUES
2                             (CHANNEL_NUMBER,3),
2                           UNMARKED_PORTION_VALUES
2                             (CHANNEL_NUMBER,2),
2                           UNMARKED_PORTION_VALUES
2                             (CHANNEL_NUMBER,1),
2                           SMOOTHNESS_CRITERIA
2                             (CHANNEL_NUMBER),
2                           MAXIMUM_SMOOTH_JUMP
2                             (CHANNEL_NUMBER),
2                           UNMARKED_PORTION_SMOOTH)

      IF ((.NOT. UNMARKED_PORTION_SMOOTH)
2         .OR.
2        (UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)
2          .GE. 320)) THEN
        UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)=0
        ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)=ROUGH_VALUE_PARAMETER
        ERROR_END(CHANNEL_NUMBER)=NOT_AN_END
      ELSE IF (ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)
2               .GT. 0)THEN
        ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)=
2       ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)-1
        ERROR_END(CHANNEL_NUMBER)=NOT_AN_END
      ELSE IF (ABS(CLEAN_DATA-AVERAGE_VALUE)
2               .GT. LIGHT_LIMIT(CHANNEL_NUMBER))THEN
        ERROR_END(CHANNEL_NUMBER)=NOT_AN_END
      ELSE IF (ABS(CLEAN_DATA-AVERAGE_VALUE)
2               .GT. TIGHT_LIMIT(CHANNEL_NUMBER))THEN
        ERROR_END(CHANNEL_NUMBER)=UNSURE
      ELSE
        IF (POSSIBLE_ERROR_START_SAMPLE
2               (CHANNEL_NUMBER,CURRENT_POSSIBLE_ERROR
2                               (CHANNEL_NUMBER))
2             .NE. 0) THEN
          IF (SAMPLE_COUNTER1 .GT. UNMARKED_PORTION_COUNTER
2                                  (CHANNEL_NUMBER))THEN
            POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
2                                  CURRENT_POSSIBLE_ERROR
2                                    (CHANNEL_NUMBER))=
2             RECORD_NUMBER
```

```
                  POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
     2                                     CURRENT_POSSIBLE_ERROR
     2                                       (CHANNEL_NUMBER))=
     2                  SAMPLE_COUNTER1-UNMARKED_PORTION_COUNTER
     2                                  (CHANNEL_NUMBER)
                     ELSE
                  POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
     2                                     CURRENT_POSSIBLE_ERROR
     2                                       (CHANNEL_NUMBER))=
     2                  RECORD_NUMBER-1
                  POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
     2                                     CURRENT_POSSIBLE_ERROR
     2                                       (CHANNEL_NUMBER))=
     2                  320-UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)+
     2                  SAMPLE_COUNTER1
                     END IF
               CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=
     2         CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)+1
               END IF
               ERROR_END(CHANNEL_NUMBER)=CERTAIN_END
            END IF
         END IF !-END IF FROM "UNMARKED_PORTION_COUNTER .LT.5 IF"

C        DO THE FOLLOWING IF THIS IS THE END OF ERROR:
         IF (ERROR_END(CHANNEL_NUMBER) .EQ. CERTAIN_END) THEN
            ERROR_LENGTH_COUNTER(CHANNEL_NUMBER)=0
            ERROR_IN_PROGRESS=1
            IF(CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)
     2         .EQ.
     2         2) THEN
               IF (((POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,1)-
     2              POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,1))
     2             *320
     2             +POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,1)
     2             -POSSIBLE_ERROR_START_SAMPLE(CHANNEL_NUMBER,1)
     2             +1)
     2            .LT. MAXIMUM_LENGTH) THEN
                  GO TO 50
               ELSE IF (NOT_A_LONG_CONSTANT(CHANNEL_NUMBER)
     2                  .OR.
     2                  (.NOT. LONG_CONSTANT_ANALYSED(CHANNEL_NUMBER)))
     2            THEN
                  VARIABILITY = UNKNOWN_VARIABILITY
                  ENTRANCE = ENTRANCE_UNKNOWN
                  EXIT = EXIT_UNKNOWN
                  UNMARKED_CATEGORY = UNMARKED
                  ERROR_LENGTH_CATEGORY =MORE_THAN_200
                  CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=1
                  START_OF_ERROR1_REC = POSSIBLE_ERROR_START_REC
     2                                  (CHANNEL_NUMBER,1)
                  START_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_START_SAMPLE
     2                                  (CHANNEL_NUMBER,1)
                  END_OF_ERROR1_REC = POSSIBLE_ERROR_END_REC
     2                                  (CHANNEL_NUMBER,1)
                  END_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_END_SAMPLE
     2                                  (CHANNEL_NUMBER,1)
                  START_OF_ERROR2_REC = 0
                  START_OF_ERROR2_SAMPLE =0
                  END_OF_ERROR2_REC = 0
                  END_OF_ERROR2_SAMPLE = 0
                  CORRECTIVE_ACTION_CODE2 = DO_NOTHING
                  CORRECTION_VALUE2 =0
                  ERROR2_GUESS = NO_ERROR
                  GO TO 60
```

```
      ELSE IF (SWITCHED_TO_1023(CHANNEL_NUMBER))THEN
         VARIABILITY = CONSTANT_LAST_TO_1023
         ENTRANCE = ENTRANCE_UNKNOWN
         EXIT = EXIT_UNKNOWN
         UNMARKED_CATEGORY = UNMARKED
         ERROR_LENGTH_CATEGORY =MORE_THAN_200
         CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=1
         START_OF_ERROR1_REC = POSSIBLE_ERROR_START_REC
     2                         (CHANNEL_NUMBER,1)
         START_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_START_SAMPLE
     2                         (CHANNEL_NUMBER,1)
         END_OF_ERROR1_REC = POSSIBLE_ERROR_END_REC
     2                         (CHANNEL_NUMBER,1)
         END_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_END_SAMPLE
     2                         (CHANNEL_NUMBER,1)
         START_OF_ERROR2_REC = 0
         START_OF_ERROR2_SAMPLE =0
         END_OF_ERROR2_REC = 0
         END_OF_ERROR2_SAMPLE = 0
         CORRECTIVE_ACTION_CODE2 = DO_NOTHING
         CORRECTION_VALUE2 =0
         ERROR2_GUESS = NO_ERROR
         GO TO 60
      ELSE
         IF (LONG_CONSTANT_VALUE(CHANNEL_NUMBER)
     2        .EQ. 1023) THEN
            VARIABILITY=CONSTANT_AT_1023
         ELSE
            VARIABILITY = CONSTANT
         END IF
         ENTRANCE = ENTRANCE_UNKNOWN
         EXIT = EXIT_UNKNOWN
         UNMARKED_CATEGORY = UNMARKED
         ERROR_LENGTH_CATEGORY =MORE_THAN_200
         CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=1
         START_OF_ERROR1_REC = POSSIBLE_ERROR_START_REC
     2                         (CHANNEL_NUMBER,1)
         START_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_START_SAMPLE
     2                         (CHANNEL_NUMBER,1)
         END_OF_ERROR1_REC = POSSIBLE_ERROR_END_REC
     2                         (CHANNEL_NUMBER,1)
         END_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_END_SAMPLE
     2                         (CHANNEL_NUMBER,1)
         START_OF_ERROR2_REC = 0
         START_OF_ERROR2_SAMPLE =0
         END_OF_ERROR2_REC = 0
         END_OF_ERROR2_SAMPLE = 0
         CORRECTIVE_ACTION_CODE2 = DO_NOTHING
         CORRECTION_VALUE2 =0
         ERROR2_GUESS = NO_ERROR
         GO TO 60
      END IF
   ELSE
      DO ERROR_COUNTER1=1,(CURRENT_POSSIBLE_ERROR
     2                    (CHANNEL_NUMBER)-1),1
      IF (((POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
     2                             ERROR_COUNTER1)-
     2       POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                                ERROR_COUNTER1))
     2       *320
     2       +POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
     2                                  ERROR_COUNTER1)
     2       -POSSIBLE_ERROR_START_SAMPLE(CHANNEL_NUMBER,
     2                                    ERROR_COUNTER1)
```

```
2            +1)
2            .GE. MAXIMUM_LENGTH) THEN
             VARIABILITY = UNKNOWN_VARIABILITY
             ENTRANCE = ENTRANCE_UNKNOWN
             EXIT = EXIT_UNKNOWN
             ERROR_LENGTH_CATEGORY =MORE_THAN_200
             UNMARKED_CATEGORY = UNMARKED
             START_OF_ERROR1_REC = POSSIBLE_ERROR_START_REC
2                                  (CHANNEL_NUMBER,1)
             START_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_START_SAMPLE
2                                  (CHANNEL_NUMBER,1)
             END_OF_ERROR1_REC = POSSIBLE_ERROR_END_REC
2                                  (CHANNEL_NUMBER,
2                       CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)-1)
             END_OF_ERROR1_SAMPLE = POSSIBLE_ERROR_END_SAMPLE
2                                  (CHANNEL_NUMBER,
2                       CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)-1)
             CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=1
             START_OF_ERROR2_REC = 0
             START_OF_ERROR2_SAMPLE =0
             END_OF_ERROR2_REC = 0
             END_OF_ERROR2_SAMPLE = 0
             CORRECTIVE_ACTION_CODE2 = DO_NOTHING
             CORRECTION_VALUE2 =0
             ERROR2_GUESS = NO_ERROR
             GO TO 60
          END IF
          END DO
          GO TO 50
       END IF

    ELSE IF (ERROR_END(CHANNEL_NUMBER) .EQ. UNSURE) THEN
       IF(POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
2                                 CURRENT_POSSIBLE_ERROR
2                                   (CHANNEL_NUMBER))
2          .NE. 0)THEN
          IF (UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)
2              .GE. SAMPLE_COUNTER1)THEN
             POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
2                                   CURRENT_POSSIBLE_ERROR
2                                     (CHANNEL_NUMBER))=
2            RECORD_NUMBER-1
             POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
2                                      CURRENT_POSSIBLE_ERROR
2                                        (CHANNEL_NUMBER))=
2            320-UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)+
2            SAMPLE_COUNTER1
          ELSE
             POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
2                                   CURRENT_POSSIBLE_ERROR
2                                     (CHANNEL_NUMBER))=
2            RECORD_NUMBER
             POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
2                                      CURRENT_POSSIBLE_ERROR
2                                        (CHANNEL_NUMBER))=
2            SAMPLE_COUNTER1-
2            UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)
          END IF
          CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=
2         CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)+1
          POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
2                                 CURRENT_POSSIBLE_ERROR
2                                   (CHANNEL_NUMBER))=0
          POSSIBLE_ERROR_START_SAMPLE(CHANNEL_NUMBER,
```

```
     2                                  CURRENT_POSSIBLE_ERROR
     2                                    (CHANNEL_NUMBER))=0
           END IF
           CORRECTIVE_ACTION_CODE1=DO_NOTHING
           CORRECTIVE_ACTION_CODE2=DO_NOTHING
           RETURN

         ELSE
C           DO THE FOLLOWING IF THIS IS AN UNMARKED ERROR, I.E.
C           NOT THE ERROR END:
            IF(POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                                 CURRENT_POSSIBLE_ERROR
     2                                   (CHANNEL_NUMBER))
     2        .EQ. 0) THEN
              CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=
     2        CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)-1
              POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
     2                               CURRENT_POSSIBLE_ERROR
     2                                 (CHANNEL_NUMBER))=0
              POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
     2                                  CURRENT_POSSIBLE_ERROR
     2                                    (CHANNEL_NUMBER))=0
           END IF
           CORRECTIVE_ACTION_CODE1 = DO_NOTHING
           CORRECTIVE_ACTION_CODE2 = DO_NOTHING
           RETURN
         END IF
       ELSE

C        DO THE FOLLOWING IF ERROR BIT IS SET OR
C        CORRECTIVE_ACTION_CODE1 .EQ. START_UNMARKED_ERROR:
         IF (((UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)
     2         .GT. 0)
     2        .AND.
     2        (ERROR_END(CHANNEL_NUMBER) .EQ. UNSURE))
     2       .OR.
     2       (ERROR_LENGTH_COUNTER(CHANNEL_NUMBER)
     2        .EQ. 1)) THEN
           POSSIBLE_ERROR_START_REC (CHANNEL_NUMBER,
     2       CURRENT_POSSIBLE_ERROR
     2       (CHANNEL_NUMBER))=RECORD_NUMBER
           POSSIBLE_ERROR_START_SAMPLE (CHANNEL_NUMBER,
     2       CURRENT_POSSIBLE_ERROR
     2       (CHANNEL_NUMBER))=SAMPLE_COUNTER1
           ERROR_LENGTH_COUNTER(CHANNEL_NUMBER)=1
         END IF
         UNMARKED_PORTION_COUNTER(CHANNEL_NUMBER)=0
         ROUGH_VALUE_DISTANCE(CHANNEL_NUMBER)=0
         IF (ERROR_LENGTH_COUNTER
     2         (CHANNEL_NUMBER) .GE. MAXIMUM_LENGTH) THEN
C          CHECK IF IT'S A LONG CONSTANT
C          AND ACT ACCORDINGLY
           IF(NOT_A_LONG_CONSTANT(CHANNEL_NUMBER)) THEN
             CORRECTIVE_ACTION_CODE1=DO_NOTHING
             CORRECTIVE_ACTION_CODE2=DO_NOTHING
             RETURN
           ELSE IF (LONG_CONSTANT(CHANNEL_NUMBER)) THEN
             IF (CLEAN_DATA .EQ.
     2          LONG_CONSTANT_VALUE(CHANNEL_NUMBER)) THEN
               CORRECTIVE_ACTION_CODE1 = DO_NOTHING
               CORRECTIVE_ACTION_CODE2 = DO_NOTHING
               RETURN
             ELSE IF (CLEAN_DATA .EQ. 1023) THEN
               SWITCHED_TO_1023(CHANNEL_NUMBER)=.TRUE.
```

```
                  CORRECTIVE_ACTION_CODE1 = DO_NOTHING
                  CORRECTIVE_ACTION_CODE2 = DO_NOTHING
                  RETURN
               ELSE
                  CORRECTIVE_ACTION_CODE1=DO_NOTHING
                  CORRECTIVE_ACTION_CODE2=DO_NOTHING
                  NOT_A_LONG_CONSTANT(CHANNEL_NUMBER)=.TRUE.
                  RETURN
               END IF
            ELSE
C              PUT DO LOOPS CHECKING IF THE LAST
C              SEVERAL(=NUMBER_OF_ERROR_VALUES) ERRORS ARE
C              CONSTANT OR 1023
               LONG_CONSTANT_ANALYSED(CHANNEL_NUMBER)=.TRUE.
               DO COUNTER2=1,NUMBER_OF_ERROR_VALUES,1
                  IF(SAMPLE_COUNTER1 .GE. COUNTER2) THEN
                     VALUES_OF_ERROR(COUNTER2)=
     2               (DATA_BLOCK(CHANNEL_NUMBER,SAMPLE_COUNTER1+1
     2                                         -COUNTER2,
     2                           PRESENT_RECORD_INDEX)
     2                .AND. DATA_MASK)
                  ELSE
                     VALUES_OF_ERROR(COUNTER2)=
     2               (DATA_BLOCK(CHANNEL_NUMBER,321+
     2                                          SAMPLE_COUNTER1-
     2                                          COUNTER2,
     2                           PREVIOUS_INDEX
     2                             (PRESENT_RECORD_INDEX))
     2                .AND. DATA_MASK)
                  END IF
               END DO
               LONG_CONSTANT(CHANNEL_NUMBER)=.TRUE.
               LONG_CONSTANT_VALUE(CHANNEL_NUMBER)=1023
               NOT_A_LONG_CONSTANT(CHANNEL_NUMBER)=.FALSE.
               COUNTER2=1
               DO WHILE ((COUNTER2 .LE. NUMBER_OF_ERROR_VALUES)
     2                    .AND.
     2                   (LONG_CONSTANT_VALUE(CHANNEL_NUMBER)
     2                    .EQ. 1023))

                  IF (SAMPLE_COUNTER1 .GE. COUNTER2)THEN
                     DO LOCAL_COUNTER20=1,8,1
                        EIGHT_CHANNELS_VALUES(LOCAL_COUNTER20)=
     2                  (DATA_BLOCK(LOCAL_COUNTER20,
     2                              (SAMPLE_COUNTER1+1-COUNTER2),
     2                              PRESENT_RECORD_INDEX)
     2                   .AND. DATA_MASK)
                     END DO
                  ELSE
                     DO LOCAL_COUNTER20=1,8,1
                        EIGHT_CHANNELS_VALUES(LOCAL_COUNTER20)=
     2                  (DATA_BLOCK(LOCAL_COUNTER20,
     2                              (SAMPLE_COUNTER1+321-COUNTER2),
     2                              PREVIOUS_INDEX
     2                                (PRESENT_RECORD_INDEX))
     2                   .AND. DATA_MASK)
                     END DO
                  END IF
                  CALL TIME_CODE_CHECK (EIGHT_CHANNELS_VALUES,
     2                                  TIME_CODE_FLAG)

                  IF ((VALUES_OF_ERROR(COUNTER2)
     2                .NE. 1023) .AND. (.NOT. TIME_CODE_FLAG))THEN
                     LONG_CONSTANT_VALUE(CHANNEL_NUMBER)=
```

```
     2              VALUES_OF_ERROR(COUNTER2)
                 END IF
                 COUNTER2=COUNTER2+1
               END DO
               COUNTER3=1
               DO WHILE ((COUNTER3 .LE. NUMBER_OF_ERROR_VALUES)
     2                .AND.
     2                (.NOT.
     2                 NOT_A_LONG_CONSTANT(CHANNEL_NUMBER)))

                 IF (SAMPLE_COUNTER1 .GE. COUNTER3)THEN
                    DO LOCAL_COUNTER20=1,8,1
                       EIGHT_CHANNELS_VALUES(LOCAL_COUNTER20)=
     2                 (DATA_BLOCK(LOCAL_COUNTER20,
     2                             (SAMPLE_COUNTER1+1-COUNTER3),
     2                             PRESENT_RECORD_INDEX)
     2                  .AND. DATA_MASK)
                    END DO
                 ELSE
                    DO LOCAL_COUNTER20=1,8,1
                       EIGHT_CHANNELS_VALUES(LOCAL_COUNTER20)=
     2                 (DATA_BLOCK(LOCAL_COUNTER20,
     2                             (SAMPLE_COUNTER1+321-COUNTER3),
     2                             PREVIOUS_INDEX
     2                               (PRESENT_RECORD_INDEX))
     2                  .AND. DATA_MASK)
                    END DO
                 END IF
                 CALL TIME_CODE_CHECK (EIGHT_CHANNELS_VALUES,
     2                                  TIME_CODE_FLAG)

                 IF((VALUES_OF_ERROR (COUNTER3)
     2               .NE. LONG_CONSTANT_VALUE(CHANNEL_NUMBER))
     2              .AND.
     2              (VALUES_OF_ERROR(COUNTER3)
     2               .NE. 1023)
     2              .AND. (.NOT. TIME_CODE_FLAG)) THEN
                    NOT_A_LONG_CONSTANT (CHANNEL_NUMBER)=.TRUE.
                 END IF
                 COUNTER3=COUNTER3+1
               END DO
               CORRECTIVE_ACTION_CODE1=DO_NOTHING
               CORRECTIVE_ACTION_CODE2=DO_NOTHING
               RETURN
            END IF
         ELSE
C           THIS IS STILL SHORT ENOUGH!
            CORRECTIVE_ACTION_CODE1=DO_NOTHING
            CORRECTIVE_ACTION_CODE2=DO_NOTHING
            RETURN
         END IF
       END IF

       RETURN
50     CONTINUE
C      FIND ENTRANCE

       IF (POSSIBLE_ERROR_START_SAMPLE(CHANNEL_NUMBER,
     2                                 ERROR_IN_PROGRESS)
     2   .GE. 5) THEN
         ENTRANCE_BLOCK_START_REC=POSSIBLE_ERROR_START_REC
     2                           (CHANNEL_NUMBER,
     2                            ERROR_IN_PROGRESS)
         ENTRANCE_BLOCK_START_SAMPLE=POSSIBLE_ERROR_START_SAMPLE
```

```
2                                (CHANNEL_NUMBER,
2                                 ERROR_IN_PROGRESS)
2                               -4
ELSE
   ENTRANCE_BLOCK_START_REC=POSSIBLE_ERROR_START_REC
2                             (CHANNEL_NUMBER,
2                              ERROR_IN_PROGRESS)
2                            -1
   ENTRANCE_BLOCK_START_SAMPLE=POSSIBLE_ERROR_START_SAMPLE
2                                (CHANNEL_NUMBER,
2                                 ERROR_IN_PROGRESS)
2                               +316
END IF
IF (ENTRANCE_BLOCK_START_SAMPLE .LE. 316) THEN
   IF (ENTRANCE_BLOCK_START_REC .EQ. RECORD_NUMBER) THEN
      DO LOCAL_COUNTER2=0,4,1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                   (ENTRANCE_BLOCK_START_SAMPLE+
2                    LOCAL_COUNTER2),PRESENT_RECORD_INDEX)

         CALL TIME_CODE_CHECK
2               (DATA_BLOCK(1,
2                           (ENTRANCE_BLOCK_START_SAMPLE+
2                            LOCAL_COUNTER2),PRESENT_RECORD_INDEX),
2                TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_ENTRANCE_BLOCK = 1
            GO TO 52
         END IF

      END DO
   ELSE IF (ENTRANCE_BLOCK_START_REC .EQ.
2           (RECORD_NUMBER-1))THEN
      DO LOCAL_COUNTER2=0,4,1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                   (ENTRANCE_BLOCK_START_SAMPLE+
2                    LOCAL_COUNTER2),
2                   PREVIOUS_INDEX(PRESENT_RECORD_INDEX))
         CALL TIME_CODE_CHECK
2               (DATA_BLOCK(1,
2                           (ENTRANCE_BLOCK_START_SAMPLE+
2                            LOCAL_COUNTER2),
2                           PREVIOUS_INDEX
2                             (PRESENT_RECORD_INDEX)),
2                TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_ENTRANCE_BLOCK = 1
            GO TO 52
         END IF

      END DO
   ELSE
      READ(2,REC=(ENTRANCE_BLOCK_START_REC+2-FIRST_RECORD),
2          ERR=10000)
2        ((SCRATCH_BLOCK(CHANNEL_COUNTER3,SAMPLE_COUNTER3),
2          CHANNEL_COUNTER3=1,8,1),
2          SAMPLE_COUNTER3=1,320,1)
      DO LOCAL_COUNTER2=0,4,1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        SCRATCH_BLOCK(CHANNEL_NUMBER,
2                      (ENTRANCE_BLOCK_START_SAMPLE+
2                       LOCAL_COUNTER2))
```

```
        CALL TIME_CODE_CHECK
2              (SCRATCH_BLOCK(1,
2                             (ENTRANCE_BLOCK_START_SAMPLE+
2                             LOCAL_COUNTER2)),
2               TIME_CODE_FLAG)
        IF (TIME_CODE_FLAG)THEN
           POINTS_IN_ENTRANCE_BLOCK = 1
           GO TO 52
        END IF

     END DO
   END IF
ELSE
   IF(ENTRANCE_BLOCK_START_REC .EQ. (RECORD_NUMBER-1))THEN
      DO LOCAL_COUNTER2=0,(320-ENTRANCE_BLOCK_START_SAMPLE),1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                   (ENTRANCE_BLOCK_START_SAMPLE
2                    +LOCAL_COUNTER2),
2                   PREVIOUS_INDEX(PRESENT_RECORD_INDEX))

         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (ENTRANCE_BLOCK_START_SAMPLE+
2                           LOCAL_COUNTER2),
2                          PREVIOUS_INDEX
2                            (PRESENT_RECORD_INDEX)),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_ENTRANCE_BLOCK = 1
            GO TO 52
         END IF

      END DO
      DO LOCAL_COUNTER2=(321-ENTRANCE_BLOCK_START_SAMPLE),4,1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                   (ENTRANCE_BLOCK_START_SAMPLE
2                    +LOCAL_COUNTER2-320),
2                   PRESENT_RECORD_INDEX)

         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (ENTRANCE_BLOCK_START_SAMPLE+
2                           LOCAL_COUNTER2-320),
2                          PRESENT_RECORD_INDEX),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_ENTRANCE_BLOCK = 1
            GO TO 52
         END IF

      END DO
   ELSE IF (ENTRANCE_BLOCK_START_REC
2           .EQ. (RECORD_NUMBER-2))THEN
      READ(2,REC=(ENTRANCE_BLOCK_START_REC+2-FIRST_RECORD),
2          ERR=10000)
2        ((SCRATCH_BLOCK(CHANNEL_COUNTER3,SAMPLE_COUNTER3),
2          CHANNEL_COUNTER3=1,8,1),
2          SAMPLE_COUNTER3=1,320,1)
      DO LOCAL_COUNTER2=0,(320-ENTRANCE_BLOCK_START_SAMPLE),1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        SCRATCH_BLOCK(CHANNEL_NUMBER,
```

```
2                 (ENTRANCE_BLOCK_START_SAMPLE
2                  +LOCAL_COUNTER2))

        CALL TIME_CODE_CHECK
2             (SCRATCH_BLOCK(1,
2                            (ENTRANCE_BLOCK_START_SAMPLE+
2                            LOCAL_COUNTER2)),
2              TIME_CODE_FLAG)
        IF (TIME_CODE_FLAG)THEN
           POINTS_IN_ENTRANCE_BLOCK = 1
           GO TO 52
        END IF

      END DO
      DO LOCAL_COUNTER2=(321-ENTRANCE_BLOCK_START_SAMPLE),4,1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                  (ENTRANCE_BLOCK_START_SAMPLE
2                   +LOCAL_COUNTER2-320),
2                  PREVIOUS_INDEX(PRESENT_RECORD_INDEX))
         CALL TIME_CODE_CHECK
2             (DATA_BLOCK(1,
2                         (ENTRANCE_BLOCK_START_SAMPLE+
2                          LOCAL_COUNTER2-320),
2                         PREVIOUS_INDEX
2                           (PRESENT_RECORD_INDEX)),
2              TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_ENTRANCE_BLOCK = 1
            GO TO 52
         END IF

      END DO
   ELSE
      READ(2,REC=(ENTRANCE_BLOCK_START_REC+2-FIRST_RECORD),
2          ERR=10000)
2       ((SCRATCH_BLOCK(CHANNEL_COUNTER3,SAMPLE_COUNTER3),
2         CHANNEL_COUNTER3=1,8,1),
2         SAMPLE_COUNTER3=1,320,1)
      DO LOCAL_COUNTER2=0,(320-ENTRANCE_BLOCK_START_SAMPLE),1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        SCRATCH_BLOCK(CHANNEL_NUMBER,
2                      (ENTRANCE_BLOCK_START_SAMPLE
2                       +LOCAL_COUNTER2))
         CALL TIME_CODE_CHECK
2             (SCRATCH_BLOCK(1,
2                            (ENTRANCE_BLOCK_START_SAMPLE+
2                            LOCAL_COUNTER2)),
2              TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_ENTRANCE_BLOCK = 1
            GO TO 52
         END IF

      END DO
      READ(2,REC=(ENTRANCE_BLOCK_START_REC+3-FIRST_RECORD),
2          ERR=10000)
2       ((SCRATCH_BLOCK(CHANNEL_COUNTER3,SAMPLE_COUNTER3),
2         CHANNEL_COUNTER3=1,8,1),
2         SAMPLE_COUNTER3=1,320,1)
      DO LOCAL_COUNTER2=(321-ENTRANCE_BLOCK_START_SAMPLE),4,1
         ENTRANCE_BLOCK(5-LOCAL_COUNTER2)=
2        SCRATCH_BLOCK(CHANNEL_NUMBER,
2                      (ENTRANCE_BLOCK_START_SAMPLE
```

```
     2                    +LOCAL_COUNTER2-320))
              CALL TIME_CODE_CHECK
     2             (SCRATCH_BLOCK(1,
     2                           (ENTRANCE_BLOCK_START_SAMPLE+
     2                           LOCAL_COUNTER2-320)),
     2              TIME_CODE_FLAG)
              IF (TIME_CODE_FLAG)THEN
                 POINTS_IN_ENTRANCE_BLOCK = 1
                 GO TO 52
              END IF

           END DO
        END IF
     END IF
     POINTS_IN_ENTRANCE_BLOCK=1
     DO LOCAL_COUNTER3=2,5,1
        IF((ENTRANCE_BLOCK(LOCAL_COUNTER3)
     2      .AND. ERROR_MASK)
     2     .NE. 0) THEN
          GO TO 52
        ELSE
           POINTS_IN_ENTRANCE_BLOCK=POINTS_IN_ENTRANCE_BLOCK+1
        END IF
     END DO
52   CONTINUE
     IF (POINTS_IN_ENTRANCE_BLOCK .EQ. 5) THEN
        CALL SMOOTHNESS_TEST (4,
     2                        (ENTRANCE_BLOCK(5) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(4) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(3) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(2) .AND. DATA_MASK),
     2                        0,
     2                        SMOOTHNESS_CRITERIA(CHANNEL_NUMBER),
     2                        MAXIMUM_SMOOTH_JUMP(CHANNEL_NUMBER),
     2                        SMOOTHNESS_FLAG)
        IF (.NOT. SMOOTHNESS_FLAG) THEN
           ENTRANCE=ENTRANCE_UNKNOWN
           POSSIBLE_ERROR_START_SAMPLE(CHANNEL_NUMBER,
     2                                 ERROR_IN_PROGRESS)=
     2     ENTRANCE_BLOCK_START_SAMPLE
           POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                               ERROR_IN_PROGRESS)=
     2     ENTRANCE_BLOCK_START_REC
        ELSE
           CALL SMOOTHNESS_TEST (5,
     2                        (ENTRANCE_BLOCK(5) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(4) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(3) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(2) .AND. DATA_MASK),
     2                        (ENTRANCE_BLOCK(1) .AND. DATA_MASK),
     2                        SMOOTHNESS_CRITERIA(CHANNEL_NUMBER),
     2                        MAXIMUM_SMOOTH_JUMP(CHANNEL_NUMBER),
     2                        SMOOTHNESS_FLAG)
           IF (SMOOTHNESS_FLAG) THEN
              ENTRANCE=ENTERED_SMOOTHLY
           ELSE
              ENTRANCE=ENTERED_ABRUPTLY
           END IF
        END IF
     ELSE
        ENTRANCE=ENTRANCE_UNKNOWN
     END IF

C    FIND EXIT
```

```
IF(POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
2                            ERROR_IN_PROGRESS)
2  .LE. 316)THEN
   EXIT_BLOCK_END_REC=POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
2                                            ERROR_IN_PROGRESS)
   EXIT_BLOCK_END_SAMPLE=POSSIBLE_ERROR_END_SAMPLE
2                          (CHANNEL_NUMBER,
2                           ERROR_IN_PROGRESS)+4
ELSE
   EXIT_BLOCK_END_REC=POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
2                                            ERROR_IN_PROGRESS)
2                    +1
   EXIT_BLOCK_END_SAMPLE=POSSIBLE_ERROR_END_SAMPLE
2                          (CHANNEL_NUMBER,
2                           ERROR_IN_PROGRESS)-316
END IF
IF(EXIT_BLOCK_END_SAMPLE .GE. 5)THEN
   IF(EXIT_BLOCK_END_REC .EQ. RECORD_NUMBER)THEN
      DO LOCAL_COUNTER4=0,4,1
         EXIT_BLOCK(LOCAL_COUNTER4+1)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                   (EXIT_BLOCK_END_SAMPLE+LOCAL_COUNTER4-4),
2                   PRESENT_RECORD_INDEX)

         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (EXIT_BLOCK_END_SAMPLE+
2                           LOCAL_COUNTER4-4),
2                          PRESENT_RECORD_INDEX),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
   ELSE IF (EXIT_BLOCK_END_REC .EQ. (RECORD_NUMBER+1))THEN
      DO LOCAL_COUNTER4=0,4,1
         EXIT_BLOCK(LOCAL_COUNTER4+1)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                   (EXIT_BLOCK_END_SAMPLE+LOCAL_COUNTER4-4),
2                   NEXT_INDEX(PRESENT_RECORD_INDEX))

         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (EXIT_BLOCK_END_SAMPLE+
2                           LOCAL_COUNTER4-4),
2                          NEXT_INDEX
2                            (PRESENT_RECORD_INDEX)),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
   ELSE
      READ (2,REC=(EXIT_BLOCK_END_REC+2-FIRST_RECORD),
2           ERR=10000)
2        ((SCRATCH_BLOCK(CHANNEL_COUNTER4,SAMPLE_COUNTER4),
2          CHANNEL_COUNTER4=1,8,1),
2          SAMPLE_COUNTER4=1,320,1)
      DO LOCAL_COUNTER4=0,4,1
```

```
         EXIT_BLOCK(LOCAL_COUNTER4+1)=
2        SCRATCH_BLOCK(CHANNEL_NUMBER,
2                  (EXIT_BLOCK_END_SAMPLE+LOCAL_COUNTER4-4))
         CALL TIME_CODE_CHECK
2              (SCRATCH_BLOCK(1,
2                  (EXIT_BLOCK_END_SAMPLE+LOCAL_COUNTER4-4)),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
   END IF
ELSE
   IF (EXIT_BLOCK_END_REC .EQ. (RECORD_NUMBER+1))THEN
      DO LOCAL_COUNTER4=0,(EXIT_BLOCK_END_SAMPLE-1),1
         EXIT_BLOCK
2          (6-EXIT_BLOCK_END_SAMPLE+LOCAL_COUNTER4)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                  (1+LOCAL_COUNTER4),
2                  NEXT_INDEX(PRESENT_RECORD_INDEX))
         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (1+LOCAL_COUNTER4),
2                          NEXT_INDEX
2                            (PRESENT_RECORD_INDEX)),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
      DO LOCAL_COUNTER4=EXIT_BLOCK_END_SAMPLE,4,1
         EXIT_BLOCK
2          (1+LOCAL_COUNTER4-EXIT_BLOCK_END_SAMPLE)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                  (316+LOCAL_COUNTER4),
2                  PRESENT_RECORD_INDEX)
         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (316+LOCAL_COUNTER4),
2                          PRESENT_RECORD_INDEX),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
   ELSE IF(EXIT_BLOCK_END_REC .EQ. RECORD_NUMBER)THEN
      DO LOCAL_COUNTER4=0,(EXIT_BLOCK_END_SAMPLE-1),1
         EXIT_BLOCK
2          (6-EXIT_BLOCK_END_SAMPLE+LOCAL_COUNTER4)=
2        DATA_BLOCK(CHANNEL_NUMBER,
2                  (1+LOCAL_COUNTER4),
2                  PRESENT_RECORD_INDEX)
         CALL TIME_CODE_CHECK
2              (DATA_BLOCK(1,
2                          (1+LOCAL_COUNTER4),
2                          PRESENT_RECORD_INDEX),
2               TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
```

```
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
      DO LOCAL_COUNTER4=EXIT_BLOCK_END_SAMPLE,4,1
         EXIT_BLOCK
     2     (1+LOCAL_COUNTER4-EXIT_BLOCK_END_SAMPLE)=
     2     DATA_BLOCK(CHANNEL_NUMBER,
     2                (316+LOCAL_COUNTER4),
     2                PREVIOUS_INDEX(PRESENT_RECORD_INDEX))
         CALL TIME_CODE_CHECK
     2          (DATA_BLOCK(1,
     2                      (316+LOCAL_COUNTER4),
     2                      PREVIOUS_INDEX
     2                        (PRESENT_RECORD_INDEX)),
     2           TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF
      END DO
   ELSE IF(EXIT_BLOCK_END_REC .EQ. (RECORD_NUMBER-1))THEN
      DO LOCAL_COUNTER4=0,(EXIT_BLOCK_END_SAMPLE-1),1
         EXIT_BLOCK
     2     (6+LOCAL_COUNTER4-EXIT_BLOCK_END_SAMPLE)=
     2     DATA_BLOCK(CHANNEL_NUMBER,
     2                (1+LOCAL_COUNTER4),
     2                PREVIOUS_INDEX(PRESENT_RECORD_INDEX))
         CALL TIME_CODE_CHECK
     2          (DATA_BLOCK(1,
     2                      (1+LOCAL_COUNTER4),
     2                      PREVIOUS_INDEX
     2                        (PRESENT_RECORD_INDEX)),
     2           TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
      READ (2,REC=(EXIT_BLOCK_END_REC + 1 - FIRST_RECORD),
     2      ERR=10000)
     2     ((SCRATCH_BLOCK(CHANNEL_COUNTER4,SAMPLE_COUNTER4),
     2       CHANNEL_COUNTER4=1,8,1),
     2       SAMPLE_COUNTER4=1,320,1)

      DO LOCAL_COUNTER4=EXIT_BLOCK_END_SAMPLE,4,1
         EXIT_BLOCK
     2     (1+LOCAL_COUNTER4-EXIT_BLOCK_END_SAMPLE)=
     2     SCRATCH_BLOCK(CHANNEL_NUMBER,
     2                   (316+LOCAL_COUNTER4))
         CALL TIME_CODE_CHECK
     2          (SCRATCH_BLOCK(1,
     2              (316 + LOCAL_COUNTER4)),
     2           TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            POINTS_IN_EXIT_BLOCK = 1
            GO TO 54
         END IF

      END DO
   ELSE
      READ (2,REC=(EXIT_BLOCK_END_REC + 2 - FIRST_RECORD),
```

```
     2          ERR=10000)
     2        ((SCRATCH_BLOCK(CHANNEL_COUNTER4,SAMPLE_COUNTER4),
     2          CHANNEL_COUNTER4=1,8,1),
     2          SAMPLE_COUNTER4=1,320,1)
           DO LOCAL_COUNTER4=0,(EXIT_BLOCK_END_SAMPLE-1),1
              EXIT_BLOCK
     2          (6+LOCAL_COUNTER4-EXIT_BLOCK_END_SAMPLE)=
     2        SCRATCH_BLOCK(CHANNEL_NUMBER,
     2                      (1+LOCAL_COUNTER4))
              CALL TIME_CODE_CHECK
     2              (SCRATCH_BLOCK(1,
     2                  (1 + LOCAL_COUNTER4)),
     2               TIME_CODE_FLAG)
              IF (TIME_CODE_FLAG)THEN
                 POINTS_IN_EXIT_BLOCK = 1
                 GO TO 54
              END IF

           END DO
           READ (2,REC=(EXIT_BLOCK_END_REC + 1 - FIRST_RECORD),
     2          ERR=10000)
     2        ((SCRATCH_BLOCK(CHANNEL_COUNTER4,SAMPLE_COUNTER4),
     2          CHANNEL_COUNTER4=1,8,1),
     2          SAMPLE_COUNTER4=1,320,1)

           DO LOCAL_COUNTER4=EXIT_BLOCK_END_SAMPLE,4,1
              EXIT_BLOCK
     2          (1+LOCAL_COUNTER4-EXIT_BLOCK_END_SAMPLE)=
     2        SCRATCH_BLOCK(CHANNEL_NUMBER,
     2                      (316+LOCAL_COUNTER4))
              CALL TIME_CODE_CHECK
     2              (SCRATCH_BLOCK(1,
     2                  (316 + LOCAL_COUNTER4)),
     2               TIME_CODE_FLAG)
              IF (TIME_CODE_FLAG)THEN
                 POINTS_IN_EXIT_BLOCK = 1
                 GO TO 54
              END IF

           END DO
        END IF
     END IF
     POINTS_IN_EXIT_BLOCK=1
     DO LOCAL_COUNTER5=2,5,1
        IF((EXIT_BLOCK(LOCAL_COUNTER5) .AND. ERROR_MASK)
     2     .NE. 0)THEN
           GO TO 54
        ELSE
           POINTS_IN_EXIT_BLOCK=POINTS_IN_EXIT_BLOCK+1
        END IF
     END DO
54   CONTINUE
     IF (POINTS_IN_EXIT_BLOCK .EQ. 5)THEN
        CALL SMOOTHNESS_TEST (5,
     2                        (EXIT_BLOCK(1) .AND. DATA_MASK),
     2                        (EXIT_BLOCK(2) .AND. DATA_MASK),
     2                        (EXIT_BLOCK(3) .AND. DATA_MASK),
     2                        (EXIT_BLOCK(4) .AND. DATA_MASK),
     2                        (EXIT_BLOCK(5) .AND. DATA_MASK),
     2                        SMOOTHNESS_CRITERIA(CHANNEL_NUMBER),
     2                        MAXIMUM_SMOOTH_JUMP(CHANNEL_NUMBER),
     2                        SMOOTHNESS_FLAG)
        IF (SMOOTHNESS_FLAG)THEN
           EXIT=EXITED_SMOOTHLY
```

```
         ELSE
            EXIT=EXITED_ABRUPTLY
         END IF
      ELSE
         EXIT=EXIT_UNKNOWN
      END IF

C     FIND VARIABILITY

      ERROR_LENGTH=(POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
     2                                     ERROR_IN_PROGRESS)
     2             -POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                                       ERROR_IN_PROGRESS))
     2            *320
     2            +POSSIBLE_ERROR_END_SAMPLE(CHANNEL_NUMBER,
     2                                       ERROR_IN_PROGRESS)
     2            -POSSIBLE_ERROR_START_SAMPLE(CHANNEL_NUMBER,
     2                                         ERROR_IN_PROGRESS)
     2            +1
      NUMBER_OF_TIME_CODES = 0
      IF (ERROR_LENGTH .GE. MAXIMUM_LENGTH)THEN
         VARIABILITY=UNKNOWN_VARIABILITY
      ELSE
         IF(POSSIBLE_ERROR_END_REC(CHANNEL_NUMBER,
     2                             ERROR_IN_PROGRESS)
     2      .EQ.
     2      POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                               ERROR_IN_PROGRESS))THEN
            IF(POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                                  ERROR_IN_PROGRESS)
     2         .EQ. RECORD_NUMBER)THEN
               DO LOCAL_COUNTER6=POSSIBLE_ERROR_START_SAMPLE
     2                           (CHANNEL_NUMBER,
     2                            ERROR_IN_PROGRESS),
     2                          POSSIBLE_ERROR_END_SAMPLE
     2                            (CHANNEL_NUMBER,
     2                             ERROR_IN_PROGRESS),1
                  ERROR_BLOCK(LOCAL_COUNTER6+1-
     2                        POSSIBLE_ERROR_START_SAMPLE
     2                          (CHANNEL_NUMBER,
     2                           ERROR_IN_PROGRESS))=
     2            DATA_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6,
     2                       PRESENT_RECORD_INDEX)

                  CALL TIME_CODE_CHECK
     2                 (DATA_BLOCK(1,
     2                             LOCAL_COUNTER6,
     2                             PRESENT_RECORD_INDEX),
     2                  TIME_CODE_FLAG)
                  IF (TIME_CODE_FLAG) THEN
                     TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
     2                                  POSSIBLE_ERROR_START_SAMPLE
     2                                    (CHANNEL_NUMBER,
     2                                     ERROR_IN_PROGRESS))
     2                  = .TRUE.
                     NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
                  ELSE
                     TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
     2                                  POSSIBLE_ERROR_START_SAMPLE
     2                                    (CHANNEL_NUMBER,
     2                                     ERROR_IN_PROGRESS))
     2                  = .FALSE.
                  END IF
```

```
      END DO
    ELSE IF(POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
     2                               ERROR_IN_PROGRESS)
     2     .EQ. (RECORD_NUMBER-1))THEN
      DO LOCAL_COUNTER6=POSSIBLE_ERROR_START_SAMPLE
     2                        (CHANNEL_NUMBER,
     2                         ERROR_IN_PROGRESS),
     2                      POSSIBLE_ERROR_END_SAMPLE
     2                        (CHANNEL_NUMBER,
     2                         ERROR_IN_PROGRESS),1
        ERROR_BLOCK(LOCAL_COUNTER6+1-
     2                  POSSIBLE_ERROR_START_SAMPLE
     2                    (CHANNEL_NUMBER,
     2                     ERROR_IN_PROGRESS))=
     2     DATA_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6,
     2                PREVIOUS_INDEX(PRESENT_RECORD_INDEX))
        CALL TIME_CODE_CHECK
     2         (DATA_BLOCK(1,
     2                     LOCAL_COUNTER6,
     2                     PREVIOUS_INDEX
     2                       (PRESENT_RECORD_INDEX)),
     2           TIME_CODE_FLAG)
        IF (TIME_CODE_FLAG) THEN
          TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
     2                           POSSIBLE_ERROR_START_SAMPLE
     2                             (CHANNEL_NUMBER,
     2                              ERROR_IN_PROGRESS))
     2       = .TRUE.
          NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
        ELSE
          TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
     2                           POSSIBLE_ERROR_START_SAMPLE
     2                             (CHANNEL_NUMBER,
     2                              ERROR_IN_PROGRESS))
     2       = .FALSE.
        END IF

      END DO
    ELSE
      READ (2,REC=(POSSIBLE_ERROR_END_REC
     2                (CHANNEL_NUMBER,ERROR_IN_PROGRESS)+2
     2              -FIRST_RECORD),ERR=10000)
     2    ((SCRATCH_BLOCK(CHANNEL_COUNTER5,SAMPLE_COUNTER5),
     2      CHANNEL_COUNTER5=1,8,1),
     2      SAMPLE_COUNTER5=1,320,1)
      DO LOCAL_COUNTER6=POSSIBLE_ERROR_START_SAMPLE
     2                        (CHANNEL_NUMBER,
     2                         ERROR_IN_PROGRESS),
     2                      POSSIBLE_ERROR_END_SAMPLE
     2                        (CHANNEL_NUMBER,
     2                         ERROR_IN_PROGRESS),1
        ERROR_BLOCK(LOCAL_COUNTER6+1-
     2                  POSSIBLE_ERROR_START_SAMPLE
     2                    (CHANNEL_NUMBER,
     2                     ERROR_IN_PROGRESS))=
     2     SCRATCH_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6)

        CALL TIME_CODE_CHECK
     2         (SCRATCH_BLOCK(1,
     2                        LOCAL_COUNTER6),
     2           TIME_CODE_FLAG)
        IF (TIME_CODE_FLAG) THEN
          TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
     2                           POSSIBLE_ERROR_START_SAMPLE
```

```
2                                  (CHANNEL_NUMBER,
2                                   ERROR_IN_PROGRESS))
2            = .TRUE.
             NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
          ELSE
             TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                               POSSIBLE_ERROR_START_SAMPLE
2                                 (CHANNEL_NUMBER,
2                                  ERROR_IN_PROGRESS))
2            = .FALSE.
          END IF

       END DO
    END IF
 ELSE
    IF(POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
2                               ERROR_IN_PROGRESS)
2       .EQ. (RECORD_NUMBER-1))THEN

       DO LOCAL_COUNTER6=POSSIBLE_ERROR_START_SAMPLE
2                          (CHANNEL_NUMBER,
2                           ERROR_IN_PROGRESS),
2                        320,1
          ERROR_BLOCK(LOCAL_COUNTER6+1-
2                     POSSIBLE_ERROR_START_SAMPLE
2                       (CHANNEL_NUMBER,
2                        ERROR_IN_PROGRESS))=
2         DATA_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6,
2                    PREVIOUS_INDEX(PRESENT_RECORD_INDEX))
          CALL TIME_CODE_CHECK
2               (DATA_BLOCK(1,
2                           LOCAL_COUNTER6,
2                           PREVIOUS_INDEX
2                             (PRESENT_RECORD_INDEX)),
2                TIME_CODE_FLAG)
          IF (TIME_CODE_FLAG) THEN
             TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                               POSSIBLE_ERROR_START_SAMPLE
2                                 (CHANNEL_NUMBER,
2                                  ERROR_IN_PROGRESS))
2            = .TRUE.
             NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
          ELSE
             TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                               POSSIBLE_ERROR_START_SAMPLE
2                                 (CHANNEL_NUMBER,
2                                  ERROR_IN_PROGRESS))
2            = .FALSE.
          END IF

       END DO

       DO LOCAL_COUNTER6=1,POSSIBLE_ERROR_END_SAMPLE
2                          (CHANNEL_NUMBER,
2                           ERROR_IN_PROGRESS),1
          ERROR_BLOCK(LOCAL_COUNTER6+321-
2                     POSSIBLE_ERROR_START_SAMPLE
2                       (CHANNEL_NUMBER,
2                        ERROR_IN_PROGRESS))=
2         DATA_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6,
2                    PRESENT_RECORD_INDEX)
          CALL TIME_CODE_CHECK
2               (DATA_BLOCK(1,
2                           LOCAL_COUNTER6,
```

```
2                          PRESENT_RECORD_INDEX),
2                TIME_CODE_FLAG)
            IF (TIME_CODE_FLAG) THEN
               TIME_CODE_IN_ERROR(LOCAL_COUNTER6+321-
2                             POSSIBLE_ERROR_START_SAMPLE
2                               (CHANNEL_NUMBER,
2                                ERROR_IN_PROGRESS))
2              = .TRUE.
               NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
            ELSE
               TIME_CODE_IN_ERROR(LOCAL_COUNTER6+321-
2                             POSSIBLE_ERROR_START_SAMPLE
2                               (CHANNEL_NUMBER,
2                                ERROR_IN_PROGRESS))
2              = .FALSE.
            END IF

         END DO

      ELSE IF(POSSIBLE_ERROR_START_REC(CHANNEL_NUMBER,
2                                    ERROR_IN_PROGRESS)
2        .EQ. (RECORD_NUMBER-2))THEN
         READ (2,REC=(POSSIBLE_ERROR_START_REC
2                     (CHANNEL_NUMBER,ERROR_IN_PROGRESS)+2
2                   -FIRST_RECORD),ERR=10000)
2          ((SCRATCH_BLOCK(CHANNEL_COUNTER5,SAMPLE_COUNTER5),
2            CHANNEL_COUNTER5=1,8,1),
2            SAMPLE_COUNTER5=1,320,1)

         DO LOCAL_COUNTER6=POSSIBLE_ERROR_START_SAMPLE
2                          (CHANNEL_NUMBER,
2                           ERROR_IN_PROGRESS),
2                        320,1
            ERROR_BLOCK(LOCAL_COUNTER6+1-
2                       POSSIBLE_ERROR_START_SAMPLE
2                         (CHANNEL_NUMBER,
2                          ERROR_IN_PROGRESS))=
2           SCRATCH_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6)

            CALL TIME_CODE_CHECK
2                 (SCRATCH_BLOCK(1,
2                                LOCAL_COUNTER6),
2                  TIME_CODE_FLAG)
            IF (TIME_CODE_FLAG) THEN
               TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                             POSSIBLE_ERROR_START_SAMPLE
2                               (CHANNEL_NUMBER,
2                                ERROR_IN_PROGRESS))
2              = .TRUE.
               NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
            ELSE
               TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                             POSSIBLE_ERROR_START_SAMPLE
2                               (CHANNEL_NUMBER,
2                                ERROR_IN_PROGRESS))
2              = .FALSE.
            END IF

         END DO
         DO LOCAL_COUNTER6=1,POSSIBLE_ERROR_END_SAMPLE
2                          (CHANNEL_NUMBER,
2                           ERROR_IN_PROGRESS),1
            ERROR_BLOCK(LOCAL_COUNTER6+321-
2                       POSSIBLE_ERROR_START_SAMPLE
```

```
2                     (CHANNEL_NUMBER,
2                      ERROR_IN_PROGRESS))=
2          DATA_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6,
2                     PREVIOUS_INDEX(PRESENT_RECORD_INDEX))

           CALL TIME_CODE_CHECK
2                (DATA_BLOCK(1,
2                            LOCAL_COUNTER6,
2                            PREVIOUS_INDEX
2                              (PRESENT_RECORD_INDEX)),
2                 TIME_CODE_FLAG)
           IF (TIME_CODE_FLAG) THEN
              TIME_CODE_IN_ERROR(LOCAL_COUNTER6+321-
2                                POSSIBLE_ERROR_START_SAMPLE
2                                  (CHANNEL_NUMBER,
2                                   ERROR_IN_PROGRESS))
2             = .TRUE.
              NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
           ELSE
              TIME_CODE_IN_ERROR(LOCAL_COUNTER6+321-
2                                POSSIBLE_ERROR_START_SAMPLE
2                                  (CHANNEL_NUMBER,
2                                   ERROR_IN_PROGRESS))
2             = .FALSE.
           END IF

        END DO
     ELSE
        READ (2,REC=(POSSIBLE_ERROR_START_REC
2                     (CHANNEL_NUMBER,ERROR_IN_PROGRESS)+2
2                    -FIRST_RECORD),ERR=10000)
2         ((SCRATCH_BLOCK(CHANNEL_COUNTER5,SAMPLE_COUNTER5),
2           CHANNEL_COUNTER5=1,8,1),
2           SAMPLE_COUNTER5=1,320,1)

        DO LOCAL_COUNTER6=POSSIBLE_ERROR_START_SAMPLE
2                         (CHANNEL_NUMBER,
2                          ERROR_IN_PROGRESS),
2                        320,1
           ERROR_BLOCK(LOCAL_COUNTER6+1-
2                      POSSIBLE_ERROR_START_SAMPLE
2                        (CHANNEL_NUMBER,
2                         ERROR_IN_PROGRESS))=
2          SCRATCH_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6)
           CALL TIME_CODE_CHECK
2               (SCRATCH_BLOCK(1,
2                              LOCAL_COUNTER6),
2                TIME_CODE_FLAG)
           IF (TIME_CODE_FLAG) THEN
              TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                                POSSIBLE_ERROR_START_SAMPLE
2                                  (CHANNEL_NUMBER,
2                                   ERROR_IN_PROGRESS))
2             = .TRUE.
              NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
           ELSE
              TIME_CODE_IN_ERROR(LOCAL_COUNTER6+1-
2                                POSSIBLE_ERROR_START_SAMPLE
2                                  (CHANNEL_NUMBER,
2                                   ERROR_IN_PROGRESS))
2             = .FALSE.
           END IF

        END DO
```

```
      READ (2,REC=(POSSIBLE_ERROR_END_REC
     2                (CHANNEL_NUMBER,ERROR_IN_PROGRESS)+2
     2              -FIRST_RECORD),ERR=10000)
     2    ((SCRATCH_BLOCK(CHANNEL_COUNTER5,SAMPLE_COUNTER5),
     2      CHANNEL_COUNTER5=1,8,1),
     2      SAMPLE_COUNTER5=1,320,1)

      DO LOCAL_COUNTER6=1,POSSIBLE_ERROR_END_SAMPLE
     2                    (CHANNEL_NUMBER,
     2                     ERROR_IN_PROGRESS),1
        ERROR_BLOCK(LOCAL_COUNTER6+321-
     2                POSSIBLE_ERROR_START_SAMPLE
     2                  (CHANNEL_NUMBER,
     2                   ERROR_IN_PROGRESS))=
     2    SCRATCH_BLOCK(CHANNEL_NUMBER,LOCAL_COUNTER6)

        CALL TIME_CODE_CHECK
     2         (SCRATCH_BLOCK(1,
     2                        LOCAL_COUNTER6),
     2          TIME_CODE_FLAG)
        IF (TIME_CODE_FLAG) THEN
          TIME_CODE_IN_ERROR(LOCAL_COUNTER6+321-
     2                          POSSIBLE_ERROR_START_SAMPLE
     2                            (CHANNEL_NUMBER,
     2                             ERROR_IN_PROGRESS))
     2      = .TRUE.
          NUMBER_OF_TIME_CODES = NUMBER_OF_TIME_CODES +1
        ELSE
          TIME_CODE_IN_ERROR(LOCAL_COUNTER6+321-
     2                          POSSIBLE_ERROR_START_SAMPLE
     2                            (CHANNEL_NUMBER,
     2                             ERROR_IN_PROGRESS))
     2      = .FALSE.
        END IF

      END DO
    END IF
  END IF

  TIME_SHIFT_INDEX = 0
  ERROR_LENGTH = ERROR_LENGTH - NUMBER_OF_TIME_CODES

  DO LOCAL_COUNTER7=1,ERROR_LENGTH,1
    DO WHILE (TIME_CODE_IN_ERROR(LOCAL_COUNTER7
     2                             + TIME_SHIFT_INDEX)
     2          .AND.
     2          (TIME_SHIFT_INDEX .LT. NUMBER_OF_TIME_CODES))
      TIME_SHIFT_INDEX = TIME_SHIFT_INDEX + 1
    END DO

    CLEAN_ERROR_BLOCK(LOCAL_COUNTER7)=
     2  ERROR_BLOCK(LOCAL_COUNTER7 +
     2              TIME_SHIFT_INDEX) .AND. DATA_MASK
  END DO

  POSSIBLE_SHIFTED_ERROR = .TRUE.
  CORRECTION_VALUE1=0
  START_UNSMOOTH=0
  ENTIRELY_CONSTANT=.TRUE.
  BEGINNING_OF_ERROR_SMOOTH=.TRUE.
  VARIABILITY_COUNTER=1
  VARIABILITY=SMOOTH_CONTINUOUS
  IF (ENTRANCE .EQ. ENTERED_SMOOTHLY)THEN
```

```
      CONTINUE
     ELSE IF ((ABS(4*CLEAN_ERROR_BLOCK(1)-AVERAGE_VALUE)
    2           .LE. TIGHT_LIMIT(CHANNEL_NUMBER))
    2          .AND. (4*CLEAN_ERROR_BLOCK(1) .LE. 1023)) THEN
        CLEAN_ERROR_BLOCK(1)=4*CLEAN_ERROR_BLOCK(1)
        CORRECTION_VALUE1=CLEAN_ERROR_BLOCK(1)

     ELSE IF ((ABS(2*CLEAN_ERROR_BLOCK(1)-AVERAGE_VALUE)
    2           .LE. TIGHT_LIMIT(CHANNEL_NUMBER))
    2          .AND. (2*CLEAN_ERROR_BLOCK(1) .LE. 1023)) THEN
        CLEAN_ERROR_BLOCK(1)=2*CLEAN_ERROR_BLOCK(1)
        CORRECTION_VALUE1=CLEAN_ERROR_BLOCK(1)

     ELSE IF ((ABS(CLEAN_ERROR_BLOCK(1)/2-AVERAGE_VALUE)
    2           .LE. TIGHT_LIMIT(CHANNEL_NUMBER))
    2          .AND. (CLEAN_ERROR_BLOCK(1)/2 .LE. 1023)) THEN
        CLEAN_ERROR_BLOCK(1)=CLEAN_ERROR_BLOCK(1)/2
        CORRECTION_VALUE1=CLEAN_ERROR_BLOCK(1)
     ELSE
        POSSIBLE_SHIFTED_ERROR =.FALSE.
     END IF

     DO WHILE ((BEGINNING_OF_ERROR_SMOOTH)
    2           .AND.
    2           (VARIABILITY_COUNTER .LT. (ERROR_LENGTH-1)))
        VARIABILITY_COUNTER=VARIABILITY_COUNTER+1
        ALREADY_SHIFTED=.FALSE.
        IF (CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
    2       .EQ. CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER-1))THEN
           CONTINUE
        ELSE
           IF(VARIABILITY_COUNTER .GE. 5)THEN
              SMOOTHNESS_TEST_INDEX=5
           ELSE
              SMOOTHNESS_TEST_INDEX=VARIABILITY_COUNTER
           END IF
56         CONTINUE
           CALL SMOOTHNESS_TEST (SMOOTHNESS_TEST_INDEX,
    2                            CLEAN_ERROR_BLOCK
    2                               (VARIABILITY_COUNTER),
    2                            CLEAN_ERROR_BLOCK
    2                               (VARIABILITY_COUNTER-1),
    2                            CLEAN_ERROR_BLOCK
    2                               (VARIABILITY_COUNTER-2),
    2                            CLEAN_ERROR_BLOCK
    2                               (VARIABILITY_COUNTER-3),
    2                            CLEAN_ERROR_BLOCK
    2                               (VARIABILITY_COUNTER-4),
    2                            SMOOTHNESS_CRITERIA
    2                               (CHANNEL_NUMBER),
    2                            MAXIMUM_SMOOTH_JUMP
    2                               (CHANNEL_NUMBER),
    2                            BEGINNING_OF_ERROR_SMOOTH)

           IF(BEGINNING_OF_ERROR_SMOOTH)THEN
              ENTIRELY_CONSTANT=.FALSE.
           ELSE IF ((.NOT. ALREADY_SHIFTED) .AND.
    2               POSSIBLE_SHIFTED_ERROR .AND.
    2               (IIABS(4*CLEAN_ERROR_BLOCK
    2                         (VARIABILITY_COUNTER)-
    2                   CLEAN_ERROR_BLOCK
    2                      (VARIABILITY_COUNTER-1))
    2                .LT.
    2                IIABS(CLEAN_ERROR_BLOCK
```

```
2                        (VARIABILITY_COUNTER)-
2                     CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER-1))))THEN
            ALREADY_SHIFTED=.TRUE.
            ORIGINAL_ERROR_BLOCK_VALUE=
2           CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
            CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2           4*CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
            VARIABILITY=SEVERAL_SHIFTED
            GO TO 56
         ELSE IF ((.NOT. ALREADY_SHIFTED) .AND.
2                POSSIBLE_SHIFTED_ERROR .AND.
2                (IIABS(CLEAN_ERROR_BLOCK
2                       (VARIABILITY_COUNTER)/2-
2                     CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER-1))
2                 .LT.
2                 IIABS(CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER)-
2                     CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER-1))))THEN
            ALREADY_SHIFTED=.TRUE.
            ORIGINAL_ERROR_BLOCK_VALUE=
2           CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
            CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2           CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)/2
            VARIABILITY=SEVERAL_SHIFTED
            GO TO 56
         ELSE IF ((.NOT. ALREADY_SHIFTED) .AND.
2                POSSIBLE_SHIFTED_ERROR .AND.
2                (IIABS(2*CLEAN_ERROR_BLOCK
2                          (VARIABILITY_COUNTER)-
2                     CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER-1))
2                 .LT.
2                 IIABS(CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER)-
2                     CLEAN_ERROR_BLOCK
2                        (VARIABILITY_COUNTER-1))))THEN
            ALREADY_SHIFTED=.TRUE.
            ORIGINAL_ERROR_BLOCK_VALUE=
2           CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
            CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2           2*CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
            VARIABILITY=SEVERAL_SHIFTED
            GO TO 56
         ELSE IF ((ENTRANCE .EQ. ENTERED_SMOOTHLY)
2                 .OR.
2                 (EXIT .EQ. EXITED_SMOOTHLY))THEN
            START_UNSMOOTH=VARIABILITY_COUNTER
            VARIABILITY=CONSTANT_PLUS_ENDS
            BEGINNING_OF_ERROR_SMOOTH=.FALSE.
            IF(ALREADY_SHIFTED)THEN
               CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2              ORIGINAL_ERROR_BLOCK_VALUE
            END IF
         ELSE
            VARIABILITY=UNKNOWN_VARIABILITY
            BEGINNING_OF_ERROR_SMOOTH=.FALSE.
            IF(ALREADY_SHIFTED)THEN
               CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2              ORIGINAL_ERROR_BLOCK_VALUE
            END IF
         END IF
```

```
      END IF
    END DO
    IF (ERROR_LENGTH .EQ. 1)THEN
       VARIABILITY=CONSTANT
       GO TO 57
    ELSE IF (BEGINNING_OF_ERROR_SMOOTH)THEN
       IF(ENTIRELY_CONSTANT)THEN
          IF (CLEAN_ERROR_BLOCK(ERROR_LENGTH)
     2         .EQ.
     2         CLEAN_ERROR_BLOCK(ERROR_LENGTH-1))THEN
             VARIABILITY=CONSTANT
          ELSE
             IF(ERROR_LENGTH .GE. 5) THEN
                CALL SMOOTHNESS_TEST(5,
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-1),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-2),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-3),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-4),
     2                               SMOOTHNESS_CRITERIA
     2                                 (CHANNEL_NUMBER),
     2                               MAXIMUM_SMOOTH_JUMP
     2                                 (CHANNEL_NUMBER),
     2                               SMOOTHNESS_FLAG)
             ELSE
                CALL SMOOTHNESS_TEST(ERROR_LENGTH,
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-1),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-2),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-3),
     2                               CLEAN_ERROR_BLOCK
     2                                 (ERROR_LENGTH-4),
     2                               SMOOTHNESS_CRITERIA
     2                                 (CHANNEL_NUMBER),
     2                               MAXIMUM_SMOOTH_JUMP
     2                                 (CHANNEL_NUMBER),
     2                               SMOOTHNESS_FLAG)
             END IF
             IF(SMOOTHNESS_FLAG)THEN
                VARIABILITY=CONSTANT_LAST_VALUE
             ELSE
                VARIABILITY=CONSTANT_PLUS_ENDS
             END IF
          END IF
       ELSE
          IF(ERROR_LENGTH .GE. 5) THEN
             CALL SMOOTHNESS_TEST(5,
     2                            CLEAN_ERROR_BLOCK
     2                              (ERROR_LENGTH),
     2                            CLEAN_ERROR_BLOCK
     2                              (ERROR_LENGTH-1),
     2                            CLEAN_ERROR_BLOCK
     2                              (ERROR_LENGTH-2),
     2                            CLEAN_ERROR_BLOCK
     2                              (ERROR_LENGTH-3),
```

```
     2                                CLEAN_ERROR_BLOCK
     2                                  (ERROR_LENGTH-4),
     2                                SMOOTHNESS_CRITERIA
     2                                  (CHANNEL_NUMBER),
     2                                MAXIMUM_SMOOTH_JUMP
     2                                  (CHANNEL_NUMBER),
     2                                SMOOTHNESS_FLAG)
              ELSE
                 CALL SMOOTHNESS_TEST(ERROR_LENGTH,
     2                                CLEAN_ERROR_BLOCK
     2                                  (ERROR_LENGTH),
     2                                CLEAN_ERROR_BLOCK
     2                                  (ERROR_LENGTH-1),
     2                                CLEAN_ERROR_BLOCK
     2                                  (ERROR_LENGTH-2),
     2                                CLEAN_ERROR_BLOCK
     2                                  (ERROR_LENGTH-3),
     2                                CLEAN_ERROR_BLOCK
     2                                  (ERROR_LENGTH-4),
     2                                SMOOTHNESS_CRITERIA
     2                                  (CHANNEL_NUMBER),
     2                                MAXIMUM_SMOOTH_JUMP
     2                                  (CHANNEL_NUMBER),
     2                                SMOOTHNESS_FLAG)
              END IF
              IF (SMOOTHNESS_FLAG)THEN
                 CONTINUE
              ELSE
                 VARIABILITY=CONSTANT_PLUS_ENDS
              END IF
           END IF
        ELSE IF (START_UNSMOOTH .EQ. 0) THEN
           CONTINUE
        ELSE
           END_UNSMOOTH=ERROR_LENGTH
           END_OF_ERROR_SMOOTH=.TRUE.
           VARIABILITY_COUNTER=ERROR_LENGTH
           DO WHILE ((END_OF_ERROR_SMOOTH) .AND.
     2               (VARIABILITY_COUNTER .GT. START_UNSMOOTH))
              END_UNSMOOTH=END_UNSMOOTH-1
              VARIABILITY_COUNTER=VARIABILITY_COUNTER-1
              ALREADY_SHIFTED=.FALSE.
              IF(CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER) .EQ.
     2           CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER+1))THEN
                 CONTINUE
              ELSE
                 IF((ERROR_LENGTH+1-VARIABILITY_COUNTER) .GE. 5)
     2             THEN
                    SMOOTHNESS_TEST_INDEX=5
                 ELSE
                    SMOOTHNESS_TEST_INDEX=
     2              ERROR_LENGTH+1-VARIABILITY_COUNTER
                 END IF
58    CONTINUE
                 CALL SMOOTHNESS_TEST (SMOOTHNESS_TEST_INDEX,
     2                                  CLEAN_ERROR_BLOCK
     2                                    (VARIABILITY_COUNTER),
     2                                  CLEAN_ERROR_BLOCK
     2                                    (VARIABILITY_COUNTER+1),
     2                                  CLEAN_ERROR_BLOCK
     2                                    (VARIABILITY_COUNTER+2),
     2                                  CLEAN_ERROR_BLOCK
     2                                    (VARIABILITY_COUNTER+3),
     2                                  CLEAN_ERROR_BLOCK
```

```
2                                (VARIABILITY_COUNTER+4),
2                              SMOOTHNESS_CRITERIA
2                                (CHANNEL_NUMBER),
2                              MAXIMUM_SMOOTH_JUMP
2                                (CHANNEL_NUMBER),
2                              END_OF_ERROR_SMOOTH)

            IF(END_OF_ERROR_SMOOTH)THEN
               CONTINUE
            ELSE IF ((.NOT. ALREADY_SHIFTED)
2                    .AND.
2                    POSSIBLE_SHIFTED_ERROR .AND.
2                    (IIABS(4*CLEAN_ERROR_BLOCK
2                              (VARIABILITY_COUNTER)-
2                             CLEAN_ERROR_BLOCK
2                               (VARIABILITY_COUNTER+1))
2                     .LT.
2                    IIABS(CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER)-
2                           CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER+1))))THEN
               ALREADY_SHIFTED=.TRUE.
               ORIGINAL_ERROR_BLOCK_VALUE=
2              CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
               CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2              4*CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
               GO TO 58

            ELSE IF ((.NOT. ALREADY_SHIFTED)
2                    .AND.
2                    POSSIBLE_SHIFTED_ERROR .AND.
2                    (IIABS(2*CLEAN_ERROR_BLOCK
2                              (VARIABILITY_COUNTER)-
2                             CLEAN_ERROR_BLOCK
2                               (VARIABILITY_COUNTER+1))
2                     .LT.
2                    IIABS(CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER)-
2                           CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER+1))))THEN
               ALREADY_SHIFTED=.TRUE.
               ORIGINAL_ERROR_BLOCK_VALUE=
2              CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
               CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2              2*CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
               GO TO 58

            ELSE IF ((.NOT. ALREADY_SHIFTED)
2                    .AND.
2                    POSSIBLE_SHIFTED_ERROR .AND.
2                    (IIABS(CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER)/2-
2                           CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER+1))
2                     .LT.
2                    IIABS(CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER)-
2                           CLEAN_ERROR_BLOCK
2                             (VARIABILITY_COUNTER+1))))THEN
               ALREADY_SHIFTED=.TRUE.
               ORIGINAL_ERROR_BLOCK_VALUE=
2              CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
               CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
2              CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)/2
```

```
                  GO TO 58

               ELSE
                  IF(ALREADY_SHIFTED)THEN
                     CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)=
     2               ORIGINAL_ERROR_BLOCK_VALUE
                  END IF

                  IF(CLEAN_ERROR_BLOCK(VARIABILITY_COUNTER)
     2               .NE.
     2               CLEAN_ERROR_BLOCK(START_UNSMOOTH))THEN
                     VARIABILITY=SEVERAL_CONST_PLUS_ENDS
                     END_OF_ERROR_SMOOTH=.FALSE.
                  ELSE
                     VARIABILITY_COUNTER=START_UNSMOOTH
                     VARIABILITY=CONSTANT_PLUS_ENDS
                     DO WHILE((VARIABILITY_COUNTER .GE.
     2                        START_UNSMOOTH)
     2                       .AND.
     2                       (VARIABILITY_COUNTER .LE.
     2                        END_UNSMOOTH)
     2                       .AND.
     2                       (VARIABILITY
     2                        .EQ. CONSTANT_PLUS_ENDS))
                        VARIABILITY_COUNTER=
     2                  VARIABILITY_COUNTER+1
                        IF(CLEAN_ERROR_BLOCK
     2                       (VARIABILITY_COUNTER)
     2                     .NE.
     2                     CLEAN_ERROR_BLOCK
     2                       (VARIABILITY_COUNTER-1))
     2                     THEN
                           VARIABILITY=SEVERAL_CONST_PLUS_ENDS
                        END IF
                     END DO
                     END_OF_ERROR_SMOOTH=.FALSE.
                  END IF
               END IF
            END IF
         END DO
      END IF
   END IF

57    CONTINUE
C     FIND ERROR_LENGTH_TYPE

      IF (ERROR_LENGTH .LE. 20)THEN
         ERROR_LENGTH_CATEGORY=LESS_THAN_21
      ELSE IF (ERROR_LENGTH .LE. 100) THEN
         ERROR_LENGTH_CATEGORY=BETWEEN_21_AND_100
      ELSE IF (ERROR_LENGTH .LE. 199) THEN
         ERROR_LENGTH_CATEGORY=BETWEEN_101_AND_200
      ELSE
         ERROR_LENGTH_CATEGORY=MORE_THAN_200
         UNMARKED_CATEGORY = UNMARKED
         GO TO 59
      END IF

C     FIND UNMARKED_CATEGORY

      TIME_SHIFT_INDEX = 0

      DO WHILE (TIME_CODE_IN_ERROR (TIME_SHIFT_INDEX + 1)
     2         .AND.
```

```
2         (TIME_SHIFT_INDEX .LT. NUMBER_OF_TIME_CODES))
   TIME_SHIFT_INDEX = TIME_SHIFT_INDEX +1
END DO

IF ((ERROR_BLOCK(TIME_SHIFT_INDEX + 1)
2    .AND. ERROR_MASK) .EQ. 0)THEN
   UNMARKED_CATEGORY = STARTS_AS_UNMARKED
   GO TO 59
END IF

UNMARKED_CATEGORY=MARKED

DO LOCAL_COUNTER8 = 1,(ERROR_LENGTH + NUMBER_OF_TIME_CODES),1
   IF (((ERROR_BLOCK(LOCAL_COUNTER8) .AND. ERROR_MASK)
2       .EQ. 0)
2      .AND. (.NOT. TIME_CODE_IN_ERROR (LOCAL_COUNTER8)))THEN
      UNMARKED_CATEGORY=UNMARKED
      GO TO 59
   END IF
END DO

59    CONTINUE

C     SET START AND END OF ERROR1:
      START_OF_ERROR1_REC=POSSIBLE_ERROR_START_REC
      2                    (CHANNEL_NUMBER,ERROR_IN_PROGRESS)
      START_OF_ERROR1_SAMPLE=POSSIBLE_ERROR_START_SAMPLE
      2                      (CHANNEL_NUMBER,ERROR_IN_PROGRESS)
      END_OF_ERROR1_REC=POSSIBLE_ERROR_END_REC
      2                  (CHANNEL_NUMBER,ERROR_IN_PROGRESS)
      END_OF_ERROR1_SAMPLE=POSSIBLE_ERROR_END_SAMPLE
      2                    (CHANNEL_NUMBER,ERROR_IN_PROGRESS)

C     PUT CODES FOR THE SECOND ERROR IF NECESSARY

      ERROR_IN_PROGRESS=ERROR_IN_PROGRESS+1

      IF (ERROR_IN_PROGRESS .GE.
      2   CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER))THEN
         CORRECTIVE_ACTION_CODE2=DO_NOTHING
         CURRENT_POSSIBLE_ERROR(CHANNEL_NUMBER)=1
         ERROR_IN_PROGRESS=1
      ELSE
         CORRECTIVE_ACTION_CODE2=EX_UNMARKED
      END IF

60    CONTINUE
      NOT_A_LONG_CONSTANT(CHANNEL_NUMBER)=.FALSE.
      LONG_CONSTANT(CHANNEL_NUMBER)=.FALSE.
      LONG_CONSTANT_ANALYSED(CHANNEL_NUMBER)=.FALSE.
      GO TO (100,200,300,400,500,600,
      2      700,800,900,1000,1100,1200,1300,1400,1500,1600,1700),
      2        ERROR_ID_ARRAY (VARIABILITY,
      2                        ENTRANCE,
      2                        EXIT,
      2                        ERROR_LENGTH_CATEGORY,
      2                        UNMARKED_CATEGORY)

100   CONTINUE !   G_1_1 HANDLER

      ERROR1_GUESS = G_1_1
      CORRECTIVE_ACTION_CODE1 = CLEAR_FLAG
      CORRECTION_VALUE1 = 0
      RETURN
```

```
200     CONTINUE !   G_1_2 HANDLER

        ERROR1_GUESS = G_1_2
        CORRECTIVE_ACTION_CODE1 = CLEAR_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

300     CONTINUE ! G_1_3 HANDLER

        ERROR1_GUESS = G_1_3
        CORRECTIVE_ACTION_CODE1 = CLEAR_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

400     CONTINUE ! G_1_4 HANDLER

        ERROR1_GUESS = G_1_4
        CORRECTIVE_ACTION_CODE1 = REPORT
        CORRECTION_VALUE1 = 0
        RETURN

500     CONTINUE ! G_2_1 HANDLER

        ERROR1_GUESS = G_2_1
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

600     CONTINUE ! G_2_2 HANDLER

        ERROR1_GUESS = G_2_2
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 1023
        RETURN

700     CONTINUE ! G_2_3 HADLER

        ERROR1_GUESS = G_2_3
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

800     CONTINUE ! G_3_1 HANDLER

        ERROR1_GUESS = G_3_1
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

900     CONTINUE ! G_3_2 HANDLER

        ERROR1_GUESS = G_3_2
        CORRECTIVE_ACTION_CODE1 = REPORT
        CORRECTION_VALUE1 = 0
        RETURN

1000    CONTINUE ! G_3_3 HANDLER

        ERROR1_GUESS = G_3_3
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN
```

```
1100    CONTINUE ! G_3_4 HANDLER

        ERROR1_GUESS = G_3_4
        CORRECTIVE_ACTION_CODE1 = REPORT
        CORRECTION_VALUE1 = 0
        RETURN

1200    CONTINUE ! G_3_5 HANDLER

        ERROR1_GUESS = G_3_5
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        RETURN

1300    CONTINUE ! G_4_1 HANDLER

        ERROR1_GUESS = G_4_1
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

1400    CONTINUE ! G_4_2 HANDLER

        ERROR1_GUESS = G_4_2
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        RETURN

1500    CONTINUE ! G_4_3 HANDLER

        ERROR1_GUESS = G_4_3
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

1600    CONTINUE ! G_4_4 HANDLER

        ERROR1_GUESS = G_4_4
        CORRECTIVE_ACTION_CODE1 = SET_FLAG
        CORRECTION_VALUE1 = 0
        RETURN

1700    CONTINUE ! UNKNOWN ERROR  HANDLER

        ERROR1_GUESS = UNKNOWN_ERROR
        CORRECTIVE_ACTION_CODE1 = UNKNOWN_ERROR_ACTION
        CORRECTION_VALUE1 = 0
        RETURN

10000   CLOSE (UNIT=1)
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        STOP
        END
```

E.3 For_each_channel Subroutine

```
      SUBROUTINE FOR_EACH_CHANNEL (CHANNEL_NUMBER,
     2                             RECORD_NUMBER,
     2                             SAMPLE_COUNTER1,
     2                             PRESENT_RECORD_INDEX,
     2                             FIRST_RECORD,
     2                             DATA_BLOCK,
     2                             AVERAGE_VALUE,
     2                             START_OF_ERROR1_REC,
     2                             START_OF_ERROR1_SAMPLE,
     2                             END_OF_ERROR1_REC,
     2                             END_OF_ERROR1_SAMPLE,
     2                             START_OF_ERROR2_REC,
     2                             START_OF_ERROR2_SAMPLE,
     2                             END_OF_ERROR2_REC,
     2                             END_OF_ERROR2_SAMPLE,
     2                             ERROR1_GUESS,
     2                             ERROR2_GUESS,
     2                             CORRECTIVE_ACTION_CODE1,
     2                             CORRECTION_VALUE1,
     2                             CORRECTIVE_ACTION_CODE2,
     2                             CORRECTION_VALUE2)


      LOGICAL*1 RUNNING_ERROR(8)/8*.FALSE./,
     2          ERROR_FLAG(8)

      INTEGER*2 DATA_VALUE,
     2          DATA_BLOCK(8,320,3)

      INTEGER*2 CHANNEL_NUMBER,
     2          ERROR1_GUESS,
     2          ERROR2_GUESS,
     2          CORRECTIVE_ACTION_CODE1,
     2          CORRECTIVE_ACTION_CODE2,
     2          PRESENT_RECORD_INDEX

      INTEGER*2 SAMPLE_COUNTER1,
     2          ERROR_BIT,
     2          ERROR_MASK,
     2          COMPUTE_AVERAGE,
     2          START_UNMARKED_ERROR,
     2          DO_NOTHING,
     2          START_OF_ERROR1_SAMPLE,
     2          END_OF_ERROR1_SAMPLE,
     2          START_OF_ERROR2_SAMPLE,
     2          END_OF_ERROR2_SAMPLE,
     2          CORRECTION_VALUE1,
     2          CORRECTION_VALUE2

      INTEGER*4 RECORD_NUMBER,
     2          FIRST_RECORD,
     2          START_OF_ERROR1_REC,
     2          END_OF_ERROR1_REC,
     2          START_OF_ERROR2_REC,
     2          END_OF_ERROR2_REC

      REAL*4    AVERAGE_VALUE

10    FORMAT(/,' CALLING ERROR_TYPE ROUTINE ')

      PARAMETER (DO_NOTHING = 1)
```

```
      PARAMETER (COMPUTE_AVERAGE = 9)

      PARAMETER (START_UNMARKED_ERROR = 11)

      PARAMETER (ERROR_MASK = 2048)

      DATA_VALUE=DATA_BLOCK(CHANNEL_NUMBER,SAMPLE_COUNTER1,
     2                      PRESENT_RECORD_INDEX)

      ERROR_BIT = DATA_VALUE .AND. ERROR_MASK

      IF (ERROR_BIT .EQ. 0) THEN
         ERROR_FLAG(CHANNEL_NUMBER)=.FALSE.
      ELSE
         ERROR_FLAG(CHANNEL_NUMBER)=.TRUE.
         RUNNING_ERROR(CHANNEL_NUMBER)=.TRUE.
      END IF

      IF (CORRECTIVE_ACTION_CODE1 .EQ. START_UNMARKED_ERROR)THEN
         RUNNING_ERROR(CHANNEL_NUMBER) = .TRUE.
      END IF

      IF (ERROR_FLAG(CHANNEL_NUMBER) .OR.
     2   RUNNING_ERROR(CHANNEL_NUMBER)) THEN
C        WRITE(6,10)
         CALL ERROR_TYPE (CHANNEL_NUMBER,
     2                    RECORD_NUMBER,
     2                    SAMPLE_COUNTER1,
     2                    PRESENT_RECORD_INDEX,
     2                    FIRST_RECORD,
     2                    DATA_BLOCK,
     2                    AVERAGE_VALUE,
     2                    START_OF_ERROR1_REC,
     2                    START_OF_ERROR1_SAMPLE,
     2                    END_OF_ERROR1_REC,
     2                    END_OF_ERROR1_SAMPLE,
     2                    START_OF_ERROR2_REC,
     2                    START_OF_ERROR2_SAMPLE,
     2                    END_OF_ERROR2_REC,
     2                    END_OF_ERROR2_SAMPLE,
     2                    ERROR1_GUESS,
     2                    ERROR2_GUESS,
     2                    CORRECTIVE_ACTION_CODE1,
     2                    CORRECTION_VALUE1,
     2                    CORRECTIVE_ACTION_CODE2,
     2                    CORRECTION_VALUE2)

         IF (RUNNING_ERROR(CHANNEL_NUMBER))THEN
            IF ((CORRECTIVE_ACTION_CODE2 .EQ. DO_NOTHING)
     2          .AND.
     2           (CORRECTIVE_ACTION_CODE1 .NE. DO_NOTHING)) THEN
               RUNNING_ERROR(CHANNEL_NUMBER)=.FALSE.
               CORRECTIVE_ACTION_CODE2=COMPUTE_AVERAGE
            ELSE
               RUNNING_ERROR(CHANNEL_NUMBER)=.TRUE.
            END IF
         END IF
      ELSE
         CORRECTIVE_ACTION_CODE1 = DO_NOTHING
         CORRECTIVE_ACTION_CODE2=COMPUTE_AVERAGE
      END IF
      RETURN
      END
```

E.4 Clear_bit Subroutine

```
      SUBROUTINE CLEAR_BIT(RECORD_NUMBER,
     2                     START_SAMPLE,
     2                     END_SAMPLE,
     2                     CHANNEL_NUMBER)

      INTEGER*2 START_SAMPLE,
     2          END_SAMPLE,
     2          CHANNEL_NUMBER,
     2          CHANNEL_COUNTER1,
     2          SAMPLE_COUNTER1,
     2          COUNTER,
     2          CLEAR_FLAG,
     2          SCRATCH_BLOCK(8,320)

      INTEGER*4 RECORD_NUMBER

      LOGICAL*1 TIME_CODE_FLAG

      PARAMETER (CLEAR_FLAG = 63487)

      READ(2,REC=RECORD_NUMBER)
     2    ((SCRATCH_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
     2      CHANNEL_COUNTER1=1,8,1),
     2      SAMPLE_COUNTER1=1,320,1)

      DO COUNTER=START_SAMPLE,END_SAMPLE,1
         CALL TIME_CODE_CHECK (SCRATCH_BLOCK(1,COUNTER),
     2                         TIME_CODE_FLAG)
         IF (.NOT. TIME_CODE_FLAG)THEN
            SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER)=
     2      SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER) .AND. CLEAR_FLAG
         END IF
      END DO

      WRITE(2,REC=RECORD_NUMBER)
     2     ((SCRATCH_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
     2       CHANNEL_COUNTER1=1,8,1),
     2       SAMPLE_COUNTER1=1,320,1)

      RETURN
      END
```

E.5 Clear_error_flag Subroutine

```
SUBROUTINE CLEAR_ERROR_FLAG (START_RECORD,
2                            END_RECORD,
2                            START_SAMPLE,
2                            END_SAMPLE,
2                            CHANNEL_NUMBER)

INTEGER*2 START_SAMPLE,
2         END_SAMPLE,
2         CHANNEL_NUMBER

INTEGER*4 START_RECORD,
2         END_RECORD,
2         COUNTER

IF(START_RECORD .EQ. END_RECORD)THEN
   CALL CLEAR_BIT(START_RECORD,
2                 START_SAMPLE,
2                 END_SAMPLE,
2                 CHANNEL_NUMBER)
ELSE IF (START_RECORD .EQ. (END_RECORD-1))THEN
   CALL CLEAR_BIT(START_RECORD,
2                 START_SAMPLE,
2                 320,
2                 CHANNEL_NUMBER)
   CALL CLEAR_BIT(END_RECORD,
2                 1,
2                 END_SAMPLE,
2                 CHANNEL_NUMBER)

ELSE
   CALL CLEAR_BIT(START_RECORD,
2                 START_SAMPLE,
2                 320,
2                 CHANNEL_NUMBER)
   DO COUNTER=1,(END_RECORD-START_RECORD-1),1
      CALL CLEAR_BIT((COUNTER+START_RECORD),
2                    1,
2                    320,
2                    CHANNEL_NUMBER)
   END DO
   CALL CLEAR_BIT(END_RECORD,
2                 1,
2                 END_SAMPLE,
2                 CHANNEL_NUMBER)

END IF
RETURN
END
```

E.6 Rescale_error Subroutine

```
SUBROUTINE RESCALE_ERROR (START_RECORD,
2                         END_RECORD,
2                         START_SAMPLE,
2                         END_SAMPLE,
2                         CHANNEL_NUMBER,
2                         CORRECTION_VALUE,
2                         SMOOTHNESS_CRITERIA,
2                         MAXIMUM_SMOOTH_JUMP)

INTEGER*2 START_SAMPLE,
2         END_SAMPLE,
2         CHANNEL_NUMBER,
2         COUNTER,
2         RESCALE_VALUE(5)/5*0/,
2         NUMBER_OF_VALUES,
2         MAXIMUM_SMOOTH_JUMP,
2         CORRECTION_VALUE


INTEGER*4 START_RECORD,
2         END_RECORD,
2         COUNTER2

REAL*4 SMOOTHNESS_CRITERIA

DO COUNTER=1,5,1
   RESCALE_VALUE(COUNTER)=0
END DO
IF (CORRECTION_VALUE .EQ. 0)THEN
   NUMBER_OF_VALUES=0
ELSE
   NUMBER_OF_VALUES=1
   RESCALE_VALUE(NUMBER_OF_VALUES)=CORRECTION_VALUE
END IF
IF(START_RECORD .EQ. END_RECORD)THEN
   CALL RESCALE_RECORD(RESCALE_VALUE,
2                      NUMBER_OF_VALUES,
2                      START_RECORD,
2                      START_SAMPLE,
2                      END_SAMPLE,
2                      CHANNEL_NUMBER,
2                      SMOOTHNESS_CRITERIA,
2                      MAXIMUM_SMOOTH_JUMP)

ELSE IF (START_RECORD .EQ. (END_RECORD-1))THEN
   CALL RESCALE_RECORD(RESCALE_VALUE,
2                      NUMBER_OF_VALUES,
2                      START_RECORD,
2                      START_SAMPLE,
2                      320,
2                      CHANNEL_NUMBER,
2                      SMOOTHNESS_CRITERIA,
2                      MAXIMUM_SMOOTH_JUMP)

   CALL RESCALE_RECORD(RESCALE_VALUE,
2                      NUMBER_OF_VALUES,
2                      END_RECORD,
2                      1,
2                      END_SAMPLE,
2                      CHANNEL_NUMBER,
2                      SMOOTHNESS_CRITERIA,
2                      MAXIMUM_SMOOTH_JUMP)
```

```
ELSE
   CALL RESCALE_RECORD(RESCALE_VALUE,
2                      NUMBER_OF_VALUES,
2                      START_RECORD,
2                      START_SAMPLE,
2                      320,
2                      CHANNEL_NUMBER,
2                      SMOOTHNESS_CRITERIA,
2                      MAXIMUM_SMOOTH_JUMP)

   DO COUNTER2=1,(END_RECORD-START_RECORD-1),1
      CALL RESCALE_RECORD(RESCALE_VALUE,
2                         NUMBER_OF_VALUES,
2                         (COUNTER+START_RECORD),
2                         1,
2                         320,
2                         CHANNEL_NUMBER,
2                         SMOOTHNESS_CRITERIA,
2                         MAXIMUM_SMOOTH_JUMP)

   END DO
   CALL RESCALE_RECORD(RESCALE_VALUE,
2                      NUMBER_OF_VALUES,
2                      END_RECORD,
2                      1,
2                      END_SAMPLE,
2                      CHANNEL_NUMBER,
2                      SMOOTHNESS_CRITERIA,
2                      MAXIMUM_SMOOTH_JUMP)

END IF
RETURN
END
```

E.7 Rescale_record Subroutine

```
      SUBROUTINE RESCALE_RECORD(RESCALE_VALUES,
     2                          NUMBER_OF_VALUES,
     2                          RECORD_NUMBER,
     2                          START_SAMPLE,
     2                          END_SAMPLE,
     2                          CHANNEL_NUMBER,
     2                          SMOOTHNESS_CRITERIA,
     2                          MAXIMUM_SMOOTH_JUMP)


      INTEGER*2 RESCALE_VALUES(5),
     2          NUMBER_OF_VALUES,
     2          START_SAMPLE,
     2          END_SAMPLE,
     2          CHANNEL_NUMBER,
     2          MAXIMUM_SMOOTH_JUMP,
     2          SCRATCH_BLOCK(8,320),
     2          COUNTER1,
     2          COUNTER2,
     2          CHANNEL_COUNTER,
     2          SAMPLE_COUNTER,
     2          DATA_MASK,
     2          ERASE_DATA,
     2          ORIGINAL_VALUE

      PARAMETER(DATA_MASK=1023)

      PARAMETER(ERASE_DATA=64512)

      INTEGER*4 RECORD_NUMBER

      REAL*4 SMOOTHNESS_CRITERIA

      LOGICAL*1 SMOOTHNESS_FLAG,
     2          TIME_CODE_FLAG

      READ(2,REC=RECORD_NUMBER)
     2   ((SCRATCH_BLOCK(CHANNEL_COUNTER,SAMPLE_COUNTER),
     2     CHANNEL_COUNTER=1,8,1),
     2     SAMPLE_COUNTER=1,320,1)

      DO COUNTER1 = START_SAMPLE,END_SAMPLE,1
         CALL TIME_CODE_CHECK (SCRATCH_BLOCK(1,COUNTER1),
     2                         TIME_CODE_FLAG)
         IF (TIME_CODE_FLAG)THEN
            GO TO 200
         END IF
         IF(NUMBER_OF_VALUES .EQ. 0)THEN
            RESCALE_VALUES(1)=
     2      SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1) .AND. DATA_MASK
            NUMBER_OF_VALUES=1
         ELSE
            IF(NUMBER_OF_VALUES .LT. 5)THEN
               RESCALE_VALUES(NUMBER_OF_VALUES+1)=
     2         SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)
     2         .AND. DATA_MASK
               NUMBER_OF_VALUES=NUMBER_OF_VALUES+1
            ELSE
               DO COUNTER2 = 1,4,1
                  RESCALE_VALUES(COUNTER2)=
     2            RESCALE_VALUES(COUNTER2+1)
               END DO
```

```
         RESCALE_VALUES(5)=
     2   SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)
     2   .AND. DATA_MASK
         END IF
         CALL SMOOTHNESS_TEST(NUMBER_OF_VALUES,
     2                        RESCALE_VALUES(1),
     2                        RESCALE_VALUES(2),
     2                        RESCALE_VALUES(3),
     2                        RESCALE_VALUES(4),
     2                        RESCALE_VALUES(5),
     2                        SMOOTHNESS_CRITERIA,
     2                        MAXIMUM_SMOOTH_JUMP,
     2                        SMOOTHNESS_FLAG)
         IF(SMOOTHNESS_FLAG)THEN
            GO TO 200
         END IF
         ORIGINAL_VALUE = RESCALE_VALUES(NUMBER_OF_VALUES)
         RESCALE_VALUES(NUMBER_OF_VALUES)=
     2   (SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)
     2   .AND. DATA_MASK)*4
         IF (RESCALE_VALUES(NUMBER_OF_VALUES) .LT. 1024)THEN
            CALL SMOOTHNESS_TEST(NUMBER_OF_VALUES,
     2                           RESCALE_VALUES(1),
     2                           RESCALE_VALUES(2),
     2                           RESCALE_VALUES(3),
     2                           RESCALE_VALUES(4),
     2                           RESCALE_VALUES(5),
     2                           SMOOTHNESS_CRITERIA,
     2                           MAXIMUM_SMOOTH_JUMP,
     2                           SMOOTHNESS_FLAG)
            IF(SMOOTHNESS_FLAG)THEN
               SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)=
     2         (SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1) .AND.
     2          ERASE_DATA) .OR. RESCALE_VALUES(NUMBER_OF_VALUES)
               GO TO 200
            END IF
         END IF
         RESCALE_VALUES(NUMBER_OF_VALUES)=
     2   (SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)
     2   .AND. DATA_MASK)*2
         IF (RESCALE_VALUES(NUMBER_OF_VALUES) .LT. 1024)THEN
            CALL SMOOTHNESS_TEST(NUMBER_OF_VALUES,
     2                           RESCALE_VALUES(1),
     2                           RESCALE_VALUES(2),
     2                           RESCALE_VALUES(3),
     2                           RESCALE_VALUES(4),
     2                           RESCALE_VALUES(5),
     2                           SMOOTHNESS_CRITERIA,
     2                           MAXIMUM_SMOOTH_JUMP,
     2                           SMOOTHNESS_FLAG)
            IF(SMOOTHNESS_FLAG)THEN
               SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)=
     2         (SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1) .AND.
     2          ERASE_DATA) .OR. RESCALE_VALUES(NUMBER_OF_VALUES)
               GO TO 200
            END IF
         END IF
         RESCALE_VALUES(NUMBER_OF_VALUES)=
     2   (SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)
     2   .AND. DATA_MASK)/2
         CALL SMOOTHNESS_TEST(NUMBER_OF_VALUES,
     2                        RESCALE_VALUES(1),
     2                        RESCALE_VALUES(2),
     2                        RESCALE_VALUES(3),
```

```
     2                          RESCALE_VALUES(4),
     2                          RESCALE_VALUES(5),
     2                          SMOOTHNESS_CRITERIA,
     2                          MAXIMUM_SMOOTH_JUMP,
     2                          SMOOTHNESS_FLAG)
           IF(SMOOTHNESS_FLAG)THEN
              SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1)=
     2        (SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER1) .AND.
     2         ERASE_DATA) .OR. RESCALE_VALUES(NUMBER_OF_VALUES)
              GO TO 200
           END IF
           RESCALE_VALUES(NUMBER_OF_VALUES) = ORIGINAL_VALUE
        END IF
200     CONTINUE
        END DO
        WRITE(2,REC=RECORD_NUMBER)
     2    ((SCRATCH_BLOCK(CHANNEL_COUNTER,SAMPLE_COUNTER),
     2      CHANNEL_COUNTER=1,8,1),
     2      SAMPLE_COUNTER=1,320,1)
        RETURN
        END
```

E.8 Set_bit Subroutine

```
SUBROUTINE SET_BIT(RECORD_NUMBER,
2                  START_SAMPLE,
2                  END_SAMPLE,
2                  CHANNEL_NUMBER)

INTEGER*2 START_SAMPLE,
2         END_SAMPLE,
2         CHANNEL_NUMBER,
2         CHANNEL_COUNTER1,
2         SAMPLE_COUNTER1,
2         COUNTER,
2         SET_FLAG,
2         SCRATCH_BLOCK(8,320)

INTEGER*4 RECORD_NUMBER

LOGICAL*1 TIME_CODE_FLAG

PARAMETER (SET_FLAG = 2048)

READ(2,REC=RECORD_NUMBER)
2    ((SCRATCH_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
2      CHANNEL_COUNTER1=1,8,1),
2      SAMPLE_COUNTER1=1,320,1)

DO COUNTER=START_SAMPLE,END_SAMPLE,1
   CALL TIME_CODE_CHECK (SCRATCH_BLOCK(1,COUNTER),
2                        TIME_CODE_FLAG)
   IF (.NOT. TIME_CODE_FLAG)THEN
      SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER)=
2     SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER) .OR. SET_FLAG
   END IF
END DO

WRITE(2,REC=RECORD_NUMBER)
2     ((SCRATCH_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
2       CHANNEL_COUNTER1=1,8,1),
2       SAMPLE_COUNTER1=1,320,1)

RETURN
END
```

E.9 Set_error_flag Subroutine

```
SUBROUTINE SET_ERROR_FLAG (START_RECORD,
2                          END_RECORD,
2                          START_SAMPLE,
2                          END_SAMPLE,
2                          CHANNEL_NUMBER)

INTEGER*2 START_SAMPLE,
2         END_SAMPLE,
2         CHANNEL_NUMBER

INTEGER*4 START_RECORD,
2         END_RECORD,
2         COUNTER

IF(START_RECORD .EQ. END_RECORD)THEN
   CALL SET_BIT(START_RECORD,
2               START_SAMPLE,
2               END_SAMPLE,
2               CHANNEL_NUMBER)
ELSE IF (START_RECORD .EQ. (END_RECORD-1))THEN
   CALL SET_BIT(START_RECORD,
2               START_SAMPLE,
2               320,
2               CHANNEL_NUMBER)
   CALL SET_BIT(END_RECORD,
2               1,
2               END_SAMPLE,
2               CHANNEL_NUMBER)

ELSE
   CALL SET_BIT(START_RECORD,
2               START_SAMPLE,
2               320,
2               CHANNEL_NUMBER)
   DO COUNTER=1,(END_RECORD-START_RECORD-1),1
      CALL SET_BIT((COUNTER+START_RECORD),
2                  1,
2                  320,
2                  CHANNEL_NUMBER)
   END DO
   CALL SET_BIT(END_RECORD,
2               1,
2               END_SAMPLE,
2               CHANNEL_NUMBER)

END IF
RETURN
END
```

E.10 Set_to_value Subroutine

```
SUBROUTINE SET_TO_VALUE   (VALUE,
2                          START_RECORD,
2                          END_RECORD,
2                          START_SAMPLE,
2                          END_SAMPLE,
2                          CHANNEL_NUMBER)

INTEGER*2 VALUE,
2         START_SAMPLE,
2         END_SAMPLE,
2         CHANNEL_NUMBER

INTEGER*4 START_RECORD,
2         END_RECORD,
2         COUNTER

IF(START_RECORD .EQ. END_RECORD)THEN
   CALL SET_VALUE(VALUE,
2                 START_RECORD,
2                 START_SAMPLE,
2                 END_SAMPLE,
2                 CHANNEL_NUMBER)
ELSE IF (START_RECORD .EQ. (END_RECORD-1))THEN
   CALL SET_VALUE(VALUE,
2                 START_RECORD,
2                 START_SAMPLE,
2                 320,
2                 CHANNEL_NUMBER)
   CALL SET_VALUE(VALUE,
2                 END_RECORD,
2                 1,
2                 END_SAMPLE,
2                 CHANNEL_NUMBER)

ELSE
   CALL SET_VALUE(VALUE,
2                 START_RECORD,
2                 START_SAMPLE,
2                 320,
2                 CHANNEL_NUMBER)
   DO COUNTER=1,(END_RECORD-START_RECORD-1),1
      CALL SET_VALUE(VALUE,
2                    (COUNTER+START_RECORD),
2                    1,
2                    320,
2                    CHANNEL_NUMBER)
   END DO
   CALL SET_VALUE(VALUE,
2                 END_RECORD,
2                 1,
2                 END_SAMPLE,
2                 CHANNEL_NUMBER)

END IF
RETURN
END
```

E.11 Set_value Subroutine

```
SUBROUTINE SET_VALUE(VALUE,
2                    RECORD_NUMBER,
2                    START_SAMPLE,
2                    END_SAMPLE,
2                    CHANNEL_NUMBER)

INTEGER*2 VALUE,
2         START_SAMPLE,
2         END_SAMPLE,
2         CHANNEL_NUMBER,
2         CHANNEL_COUNTER1,
2         SAMPLE_COUNTER1,
2         COUNTER,
2         SCRATCH_BLOCK(8,320)

INTEGER*4 RECORD_NUMBER,
2         CLEAR_DATA_MASK

LOGICAL*1 TIME_CODE_FLAG

PARAMETER (CLEAR_DATA_MASK = 64512)

READ(2,REC=RECORD_NUMBER)
2    ((SCRATCH_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
2      CHANNEL_COUNTER1=1,8,1),
2      SAMPLE_COUNTER1=1,320,1)

DO COUNTER=START_SAMPLE,END_SAMPLE,1
   CALL TIME_CODE_CHECK (SCRATCH_BLOCK(1,COUNTER),
2                        TIME_CODE_FLAG)
   IF (.NOT. TIME_CODE_FLAG)THEN
      SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER)=
2     ((SCRATCH_BLOCK(CHANNEL_NUMBER,COUNTER) .AND.
2     CLEAR_DATA_MASK) .OR. VALUE)
   END IF
END DO

WRITE(2,REC=RECORD_NUMBER)
2     ((SCRATCH_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
2       CHANNEL_COUNTER1=1,8,1),
2       SAMPLE_COUNTER1=1,320,1)

RETURN
END
```

E.12 Smoothness_test Subroutine

```
      SUBROUTINE SMOOTHNESS_TEST(NUMBER_OF_DATA_POINTS,
     2                           X1,
     2                           X2,
     2                           X3,
     2                           X4,
     2                           X5,
     2                           SMOOTHNESS_CRITERIA,
     2                           MAXIMUM_SMOOTH_JUMP,
     2                           SMOOTHNESS_FLAG)


      INTEGER*2 NUMBER_OF_DATA_POINTS

      INTEGER*2 X1,X2,X3,X4,X5,
     2          MAXIMUM_SMOOTH_JUMP

      REAL*4  SMOOTHNESS_CRITERIA,
     2       LINEAR_FIT_DIFFERENCE1,
     2       LINEAR_FIT_DIFFERENCE2,
     2       LINEAR_FIT_DIFFERENCE3,
     2       LINEAR_FIT_DIFFERENCE4,
     2       LINEAR_FIT_DIFFERENCE5,
     2       LINEAR_FIT_OFFSET,
     2       LINEAR_FIT_SLOPE

      LOGICAL*1 SMOOTHNESS_FLAG

10    FORMAT (/,' SMOOTHNESS_TEST CALLED WITH ',I2,' POINTS')
20    FORMAT (/,' SMOOTHNESS_FLAG ',L1)

C     WRITE (6,10)NUMBER_OF_DATA_POINTS

      IF (NUMBER_OF_DATA_POINTS .LT. 5)THEN
         GO TO (30,31,32,33),NUMBER_OF_DATA_POINTS
30       SMOOTHNESS_FLAG = .TRUE.
C        WRITE(6,20)SMOOTHNESS_FLAG

         RETURN
31       CONTINUE
         IF ((IIABS(X1-X2))
     2        .LE. MAXIMUM_SMOOTH_JUMP) THEN
            SMOOTHNESS_FLAG=.TRUE.
         ELSE
            SMOOTHNESS_FLAG=.FALSE.
         END IF
C        WRITE(6,20)SMOOTHNESS_FLAG
         RETURN
32       CONTINUE
         LINEAR_FIT_DIFFERENCE1=(X1/6.0)
     2                          -(X2/3.0)
     2                          +(X3/6.0)
         LINEAR_FIT_DIFFERENCE2=-(X1/3.0)
     2                          +(2*X2/3.0)
     2                          -(X3/3.0)
         IF(((LINEAR_FIT_DIFFERENCE1**2)*2
     2       +(LINEAR_FIT_DIFFERENCE2**2))/2.0
     2       .LE. SMOOTHNESS_CRITERIA) THEN
            SMOOTHNESS_FLAG=.TRUE.
         ELSE
            SMOOTHNESS_FLAG=.FALSE.
         END IF
C        WRITE(6,20)SMOOTHNESS_FLAG
```

```
      RETURN
33    CONTINUE
      LINEAR_FIT_OFFSET=((14.0*X1+
     2                    8.0*X2+
     2                    2.0*X3-
     2                    4.0*X4)/20.0)
      LINEAR_FIT_SLOPE=((-6.0*X1-
     2                    2.0*X2+
     2                    2.0*X3+
     2                    6.0*X4)/20.0)
      LINEAR_FIT_DIFFERENCE1=X1-
     2                       LINEAR_FIT_OFFSET
      LINEAR_FIT_DIFFERENCE2=X2-
     2                       LINEAR_FIT_OFFSET-
     2                       LINEAR_FIT_SLOPE
      LINEAR_FIT_DIFFERENCE3=X3-
     2                       LINEAR_FIT_OFFSET-
     2                       2.0*LINEAR_FIT_SLOPE
      LINEAR_FIT_DIFFERENCE4=X4-
     2                       LINEAR_FIT_OFFSET-
     2                       3.0*LINEAR_FIT_SLOPE
      IF((LINEAR_FIT_DIFFERENCE1**2+
     2     LINEAR_FIT_DIFFERENCE2**2+
     2     LINEAR_FIT_DIFFERENCE3**2+
     2     LINEAR_FIT_DIFFERENCE4**2)/3.0
     2    .LE. SMOOTHNESS_CRITERIA) THEN
         SMOOTHNESS_FLAG=.TRUE.
      ELSE
         SMOOTHNESS_FLAG=.FALSE.
      END IF
    ELSE !-ELSE FROM "NUMBER_OF_DATA_POINTS .LT.5 IF"
      LINEAR_FIT_OFFSET=((X1*1.0+
     2                    X2*1.0+
     2                    X3*1.0+
     2                    X4*1.0+
     2                    X5*1.0)/5.0)
      LINEAR_FIT_SLOPE=((-1.0*X1-
     2                     .5*X2+
     2                     .5*X4+
     2                    1.0*X5)/5.0)
      LINEAR_FIT_DIFFERENCE1=X1-
     2                       LINEAR_FIT_OFFSET+
     2                       2.0*LINEAR_FIT_SLOPE
      LINEAR_FIT_DIFFERENCE2=X2-
     2                       LINEAR_FIT_OFFSET+
     2                       LINEAR_FIT_SLOPE
      LINEAR_FIT_DIFFERENCE3=X3-
     2                       LINEAR_FIT_OFFSET
      LINEAR_FIT_DIFFERENCE4=X4-
     2                       LINEAR_FIT_OFFSET-
     2                       LINEAR_FIT_SLOPE
      LINEAR_FIT_DIFFERENCE5=X5-
     2                       LINEAR_FIT_OFFSET-
     2                       2.0*LINEAR_FIT_SLOPE
      IF ((LINEAR_FIT_DIFFERENCE1**2+
     2     LINEAR_FIT_DIFFERENCE2**2+
     2     LINEAR_FIT_DIFFERENCE3**2+
     2     LINEAR_FIT_DIFFERENCE4**2+
     2     LINEAR_FIT_DIFFERENCE5**2)/4.0
     2     .LE. SMOOTHNESS_CRITERIA) THEN
         SMOOTHNESS_FLAG=.TRUE.
      ELSE
         SMOOTHNESS_FLAG=.FALSE.
      END IF
```

```
      END IF !-END IF FROM "NUMBER_OF_DATA_POINTS .LT.5 IF"
C        WRITE(6,20)SMOOTHNESS_FLAG
      RETURN
      END
```

E.13 Time_code_check Subroutine

```
SUBROUTINE TIME_CODE_CHECK (EIGHT_VALUES,
2                           TIME_CODE_FLAG)

INTEGER*2 EIGHT_VALUES (8),
2         DATA_MASK,
2         CODE_COUNTER

LOGICAL*1 TIME_CODE_FLAG
PARAMETER (DATA_MASK = 1023)

CODE_COUNTER = 0

IF ((EIGHT_VALUES(1) .AND. DATA_MASK) .EQ. 0) THEN
   CODE_COUNTER=CODE_COUNTER+1
END IF

IF ((EIGHT_VALUES(2) .AND. DATA_MASK) .EQ. 0) THEN
   CODE_COUNTER=CODE_COUNTER+1
END IF

IF ((EIGHT_VALUES(7) .AND. DATA_MASK) .EQ. 0) THEN
   CODE_COUNTER=CODE_COUNTER+1
END IF

IF ((EIGHT_VALUES(8) .AND. DATA_MASK) .EQ. 0) THEN
   CODE_COUNTER=CODE_COUNTER+1
END IF

IF (CODE_COUNTER .GE. 3) THEN
   TIME_CODE_FLAG = .TRUE.
ELSE
   TIME_CODE_FLAG = .FALSE.
END IF

RETURN
END
```

E.14 Update_data_block Subroutine

```
      SUBROUTINE UPDATE_DATA_BLOCK (DATA_BLOCK,
     2                              PRESENT_RECORD_INDEX,
     2                              RECORD_NUMBER,
     2                              ERROR_START_REC,
     2                              ERROR_END_REC)

      INTEGER*2 DATA_BLOCK(8,320,3),
     2         PRESENT_RECORD_INDEX,
     2         PREVIOUS_INDEX(3)/3,1,2/,
     2         CHANNEL_COUNTER,
     2         SAMPLE_COUNTER

      INTEGER*4 RECORD_NUMBER,
     2         ERROR_START_REC,
     2         ERROR_END_REC


      IF(ERROR_END_REC .EQ. RECORD_NUMBER) THEN
C        LOAD ONE RECORD OF DATA INTO DATA_BLOCK
         READ(2,REC=RECORD_NUMBER)
     2      ((DATA_BLOCK(CHANNEL_COUNTER,SAMPLE_COUNTER,
     2                   PRESENT_RECORD_INDEX),
     2        CHANNEL_COUNTER=1,8,1),
     2        SAMPLE_COUNTER=1,320,1)
         IF (ERROR_START_REC .NE. RECORD_NUMBER) THEN
C           LOAD ANOTHER RECORD OF DATA INTO DATA_BLOCK
            READ(2,REC=(RECORD_NUMBER-1))
     2         ((DATA_BLOCK(CHANNEL_COUNTER,SAMPLE_COUNTER,
     2                      PREVIOUS_INDEX
     2                        (PRESENT_RECORD_INDEX)),
     2           CHANNEL_COUNTER=1,8,1),
     2           SAMPLE_COUNTER=1,320,1)
         END IF
      ELSE IF(ERROR_END_REC .EQ. (RECORD_NUMBER-1))THEN
C        LOAD ONE RECORD OF DATA INTO DATA_BLOCK
         READ(2,REC=(RECORD_NUMBER-1))
     2      ((DATA_BLOCK(CHANNEL_COUNTER,SAMPLE_COUNTER,
     2                   PREVIOUS_INDEX
     2                     (PRESENT_RECORD_INDEX)),
     2        CHANNEL_COUNTER=1,8,1),
     2        SAMPLE_COUNTER=1,320,1)

      END IF
      RETURN
      END
```

Appendix F

Average

```
      PROGRAM AVERAGING
C     ------------------------------------------------
C     DECLARATIONS:
C     INTEGERS:
      INTEGER*2 ACCEPT_CHANNEL,
     2          CHANNELS_ENTERED,
     2          COUNTER1,
     2          COUNTER2,
     2          COUNTER3,
     2          DATA_IN_CHANNEL(8)/8*0/,
     2          NO_DATA,
     2          AVERAGE_OVER_N(8)/8*100/,
     2          CHANNEL_COUNTER1,
     2          CHANNEL_COUNTER2,
     2          CHANNEL_COUNTER3,
     2          SAMPLE_COUNTER1,
     2          SAMPLE_COUNTER2,
     2          DATA_MASK,
     2          ERROR_MASK,
     2          DATA_VALUE/0/,
     2          TOTAL_VALUES_COUNTER(8)/8*0/

      INTEGER*4 RECORD_NUMBER,
     2          VALUES_SUM(8)/8*0/,
     2          FIRST_RECORD/3/,
     2          LAST_RECORD/0/

C     REAL:
      REAL*8 CORRECT_VALUES_COUNTER(8)/8*0.0/,
     2       DECIMAL_TIME/0.0/
C     LOGICAL:
      LOGICAL*1 KEEP_READING/.TRUE./,
     2          OPENED_CHANNEL(8)/8*.FALSE./,
     2          TIME_CODE_FLAG

      LOGICAL*2 DATA_BLOCK(8,320)/2560*0/

C     CHARACTER:
      CHARACTER*80 INPUT_FILE,
     2             OUTPUT_FILE(8)

C     ------------------------------------------------
C     PARAMETER VALUATION:
      PARAMETER (NO_DATA = 0)
      PARAMETER (DATA_MASK = 1023)
      PARAMETER (ERROR_MASK = 2048)
```

```
C       ------------------------------------------------
C       FORMAT STATEMENTS

10      FORMAT(/,' PLEASE ENTER DATA FILE SPECIFICATIONS: ',$)
20      FORMAT(1A)
30      FORMAT(/,
        2' PLEASE ENTER OUTPUT FILE SPECIFICATIONS FOR CHANNEL ',
        2I1,': ',$)
50      FORMAT(/,' PLEASE ENTER THE CHANNEL TO BE PROCESSED.',
        2/,' ENTER -1 TO EXIT SELECTION, 0 TO SELECT ALL CHANNELS.')
60      FORMAT(/,' PLEASE ENTER THE NEXT CHANNEL: ',$)
70      FORMAT(BN,I2)
80      FORMAT(BN,I12)
90      FORMAT(/,' PLEASE ENTER THE FIRST RECORD TO BE PROCESSED.',
        2/,' (DEFAULT IS ',I1,'):',$)
91      FORMAT(/,' PLEASE ENTER THE LAST RECORD TO BE PROCESSED.',
        2/,' ENTER 0 IF YOU WANT TO PROCESS ENTIRE FILE.',
        2/,' (DEFAULT IS ENTIRE FILE):',$)
92      FORMAT(/,
        2' PLEASE ENTER THE NUMBER OF VALUES TO AVERAGE OVER FOR',
        2' CHANNEL ',I1,':',$)
93      FORMAT(' ',F10.7,'       ',F6.1)
C       ------------------------------------------------

C       READ THE NAME OF THE INPUT FILE:

        WRITE (6,10)
        READ  (5,20) INPUT_FILE

C       OPEN INPUT FILE FOR READING

        OPEN  (UNIT=9,ACCESS='DIRECT',FORM='UNFORMATTED',
        2 STATUS='OLD',READONLY,FILE=INPUT_FILE)

C       READ WHICH CHANNELS TO PROCESS

        WRITE(6,50)
        DO WHILE (KEEP_READING)
           WRITE(6,60)
           READ(5,70)ACCEPT_CHANNEL

           IF ((ACCEPT_CHANNEL .EQ. -1).AND.
        2  (CHANNELS_ENTERED .NE. 0))THEN
              KEEP_READING=.FALSE.
           END IF


           IF (ACCEPT_CHANNEL .EQ. 0)THEN
              DO COUNTER1=1,8,1
                 DATA_IN_CHANNEL(COUNTER1)=1
              END DO
              KEEP_READING=.FALSE.
           END IF

           IF ((ACCEPT_CHANNEL .GE. 1).AND.
        2  (ACCEPT_CHANNEL .LE. 8)) THEN
              CHANNELS_ENTERED=CHANNELS_ENTERED+1-DATA_IN_CHANNEL
        2     (ACCEPT_CHANNEL)
              DATA_IN_CHANNEL(ACCEPT_CHANNEL)=1
           END IF
        END DO
C       READ THE NAMES OF THE OUTPUT FILES:
```

```
      DO COUNTER1=1,8,1
         IF (DATA_IN_CHANNEL(COUNTER1) .NE. NO_DATA) THEN
            WRITE (6,30) COUNTER1
            READ  (5,20) OUTPUT_FILE(COUNTER1)
         END IF
      END DO

C     READ THE NUMBERS OF THE FIRST AND LAST RECORDS TO BE PROCESSED.

      WRITE (6,90) FIRST_RECORD
      READ  (5,80) FIRST_RECORD

      IF (FIRST_RECORD .EQ. 0)THEN
         FIRST_RECORD=3
      END IF

      DO WHILE (FIRST_RECORD .LE. 2)
         WRITE (6,90)
         READ  (5,80) FIRST_RECORD
      END DO

      WRITE(6,91)
      READ (5,80)LAST_RECORD

      DO WHILE ((LAST_RECORD .LT. FIRST_RECORD)
     2         .AND.
     2         (LAST_RECORD .NE. 0))
         WRITE(6,91)
         READ (5,80)LAST_RECORD
      END DO

C     READ THE NUMBER OF VALUES TO AVERAGE OVER:

      DO COUNTER3=1,8,1
         IF (DATA_IN_CHANNEL(COUNTER3) .NE. NO_DATA) THEN
            WRITE (6,92) COUNTER3
            READ  (5,80) AVERAGE_OVER_N(COUNTER3)
         END IF
      END DO

C     ----------------------------------------------

C     ----------------------------------------------


C     READ ''FIRST'' RECORD OF DATA:

      RECORD_NUMBER=FIRST_RECORD

      READ (9,REC=RECORD_NUMBER,ERR=1000)
     2  ((DATA_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
     2    CHANNEL_COUNTER1=1,8,1),
     2    SAMPLE_COUNTER1=1,320,1)

C     ----------------------------------------------

C     ----------------------------------------------

C     MAIN LOOP OF THE PROGRAM

100   CONTINUE

      DO SAMPLE_COUNTER2=1,320,1
         CALL TIME_CODE_CHECK(DATA_BLOCK(1,SAMPLE_COUNTER2),
```

```
2                       TIME_CODE_FLAG)
   IF(TIME_CODE_FLAG)THEN
      CALL GET_DECIMAL_TIME(DATA_BLOCK(1,SAMPLE_COUNTER2),
2                           DECIMAL_TIME)
      GO TO 200
   ELSE IF(DECIMAL_TIME .EQ. 0.0) THEN
      GO TO 200
   ELSE
      DECIMAL_TIME=DECIMAL_TIME+(1.0/360000)
   END IF
   DO CHANNEL_COUNTER2=1,8,1
      IF (DATA_IN_CHANNEL(CHANNEL_COUNTER2)
2         .EQ. NO_DATA) THEN
         GO TO 300
      END IF
      IF((DATA_BLOCK(CHANNEL_COUNTER2,
2                    SAMPLE_COUNTER2)
2          .AND. ERROR_MASK)
2         .EQ. 0)THEN
         DATA_VALUE=
2        (DATA_BLOCK(CHANNEL_COUNTER2,
2                    SAMPLE_COUNTER2)
2          .AND. DATA_MASK)
         VALUES_SUM(CHANNEL_COUNTER2)=
2        VALUES_SUM(CHANNEL_COUNTER2)+
2        DATA_VALUE
         CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)=
2        CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)+1.0
      END IF
      TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)=
2     TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)+1
      IF (TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)
2         .GE. AVERAGE_OVER_N(CHANNEL_COUNTER2))THEN
         IF ((CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2) .NE. 0)
2            .AND. OPENED_CHANNEL(CHANNEL_COUNTER2))THEN
            WRITE((CHANNEL_COUNTER2+9),93)
2                DECIMAL_TIME,
2                VALUES_SUM(CHANNEL_COUNTER2)/
2                CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)
            TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)=0
            CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)=0.0
            VALUES_SUM(CHANNEL_COUNTER2)=0
         ELSE IF ((CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)
2                  .NE. 0)
2             .AND.
2            (.NOT. OPENED_CHANNEL(CHANNEL_COUNTER2)))THEN
            OPEN(UNIT=(CHANNEL_COUNTER2+9),STATUS='NEW',
2                FILE=OUTPUT_FILE(CHANNEL_COUNTER2))
            OPENED_CHANNEL(CHANNEL_COUNTER2)=.TRUE.
            WRITE((CHANNEL_COUNTER2+9),93)
2                DECIMAL_TIME,
2                VALUES_SUM(CHANNEL_COUNTER2)/
2                CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)
            TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)=0
            CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)=0.0
            VALUES_SUM(CHANNEL_COUNTER2)=0
         ELSE IF ((CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)
2                  .EQ. 0)
2             .AND.
2            OPENED_CHANNEL(CHANNEL_COUNTER2))THEN
            CLOSE(UNIT=(CHANNEL_COUNTER2+9))
            OPENED_CHANNEL(CHANNEL_COUNTER2)=.FALSE.
            TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)=0
            CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)=0.0
```

```
                  VALUES_SUM(CHANNEL_COUNTER2)=0
               ELSE
                  TOTAL_VALUES_COUNTER(CHANNEL_COUNTER2)=0
                  CORRECT_VALUES_COUNTER(CHANNEL_COUNTER2)=0.0
                  VALUES_SUM(CHANNEL_COUNTER2)=0
               END IF
            END IF
300         CONTINUE
         END DO
200      CONTINUE
      END DO
      RECORD_NUMBER=RECORD_NUMBER+1


      READ (9,REC=RECORD_NUMBER,ERR=3000)
     2  ((DATA_BLOCK(CHANNEL_COUNTER1,SAMPLE_COUNTER1),
     2    CHANNEL_COUNTER1=1,8,1),
     2    SAMPLE_COUNTER1=1,320,1)

      IF ((RECORD_NUMBER .GT. LAST_RECORD)
     2   .AND.
     2   (LAST_RECORD .NE. 0))THEN
         GO TO 4000
      ELSE
         GO TO 100
      END IF

C     END THE PROGRAM WHEN THERE WAS NOT ''FIRST'' RECORD IN THE
C     INPUT FILE
1000  CONTINUE
      DO CHANNEL_COUNTER3=1,8,1
         IF(OPENED_CHANNEL(CHANNEL_COUNTER3))THEN
            CLOSE(UNIT=(9+CHANNEL_COUNTER3))
         END IF
      END DO
      CLOSE(UNIT=9)
      STOP

C     END THE PROGRAM WHEN THERE WAS NOT ANY MORE SPACE
C     IN THE OUTPUT FILE.
2000  CONTINUE
      DO CHANNEL_COUNTER3=1,8,1
         IF(OPENED_CHANNEL(CHANNEL_COUNTER3))THEN
            CLOSE(UNIT=(9+CHANNEL_COUNTER3))
         END IF
      END DO
      CLOSE(UNIT=9)
      STOP

C     END THE PROGRAM WHEN THERE ARE NO MORE INPUT RECORDS
3000  CONTINUE
      DO CHANNEL_COUNTER3=1,8,1
         IF(OPENED_CHANNEL(CHANNEL_COUNTER3))THEN
            CLOSE(UNIT=(9+CHANNEL_COUNTER3))
         END IF
      END DO
      CLOSE(UNIT=9)
      STOP

C     END THE PROGRAM AFTER PROCESSING THE ''LAST'' RECORD.
4000  CONTINUE
      DO CHANNEL_COUNTER3=1,8,1
         IF(OPENED_CHANNEL(CHANNEL_COUNTER3))THEN
            CLOSE(UNIT=(9+CHANNEL_COUNTER3))
```

```
   END IF
END DO
CLOSE(UNIT=9)
STOP
END
```

Appendix G

Manual_clean

```
      PROGRAM MANUAL_CLEAN
C     A LOCAL UTILITY COUNTER USED TO INPUT NUMBERS OF CHANNELS
C     TO BE PROCESSED.
      INTEGER*2 COUNTER1

C     IMPORTANT COUNTER USED FOR LOOPING THROUGH ALL CHANNELS
C     DURING ERROR PROCESSING.
      INTEGER*2 CHANNEL_COUNTER1

C     LOCAL UTILITY VARIABLE USED DURING INPUT OF CHANNEL
C     NUMBERS TO BE PROCESSED.
      INTEGER*2 ACCEPT_CHANNEL /-1/

C     ARRAY THAT SHOWS WHICH CHANNELS WERE CHOSEN FOR PROCESSING.
C     USED IN THE MAIN LOOP TO DECIDE WHETHER TO PROCESS DATA
C     FROM THAT CHANNEL OR NOT.
      INTEGER*2 DATA_IN_CHANNEL(8)/8*0/

C     A LOCAL UTILITY VARIABLE USED TO COUNT HOW MANY CHANNELS WERE
C     ENTERED FOR PROCESSING.
      INTEGER*2 CHANNELS_ENTERED /0/

C     A PARAMETER DECLARATION. THIS PARAMETER IS USED WITH THE
C     DATA_IN_CHANNEL ARRAY TO DECIDE WHETHER TO PROCESS DATA IN
C     A CHANNEL OR NOT.
      INTEGER*2 NO_DATA

C     VARIABLES THAT HOLD, RESPECTIVELY, THE FIRST SAMPLE OF
C     OF THE FIRST RECORD, AND THE LAST SAMPLE OF THE LAST RECORD
C     TO PROCESS
      INTEGER*2 START_SAMPLE,END_SAMPLE

      INTEGER*2 ACTION_CHOICE/0/,REPEAT_CODE/1/

C     VARIABLE THAT HOLDS THE NUMBER OF THE FIRST RECORD IN THE INPUT
C     FILE TO BE PROCESSED (MINIMUM IS 2)
      INTEGER*4 FIRST_RECORD

C     VARIABLE THAT HOLDS THE NUMBER OF THE LAST RECORD IN THE INPUT
C     FILE TO BE PROCESSED (MINIMUM IS FIRST_RECORD)
      INTEGER*4 LAST_RECORD

C     VARIABLE THAT HOLDS THE NAME OF THE DATA FILE.
      CHARACTER*50 INPUT_FILE

      LOGICAL*1 CLEAN_THE_DATA /.TRUE./,
```

```
     2         KEEP_READING /.TRUE./

C    SEE DECLARATION FOR EXPLANATIONS
     PARAMETER (NO_DATA = 0)
C    ------------------------------------------------

C    FORMAT STATEMENTS

10   FORMAT(/,' PLEASE ENTER DATA FILE SPECIFICATIONS: ',$)
20   FORMAT(1A)
50   FORMAT(/,' PLEASE ENTER THE CHANNEL TO BE PROCESSED.',
     2/,' ENTER -1 TO EXIT SELECTION, 0 TO SELECT ALL CHANNELS.')
60   FORMAT(/,' PLEASE ENTER THE NEXT CHANNEL: ',$)
70   FORMAT(BN,I2)
80   FORMAT(BN,I12)
90   FORMAT(/,' PLEASE ENTER THE FIRST RECORD TO BE PROCESSED: ',$)
91   FORMAT(/,' PLEASE ENTER THE LAST RECORD TO BE PROCESSED: ',$)
92   FORMAT(/,' PLEASE ENTER THE STARTING SAMPLE TO BE PROCESSED: ',$)
93   FORMAT(/,' PLEASE ENTER THE ENDING SAMPLE TO BE PROCESSED: ',$)
94   FORMAT(/,' ENTER 1 TO CLEAN THE ERROR FLAG, ANY OTHER NUMBER',
     2' TO SET IT: ',$)
95   FORMAT(/,' ENTER 1 TO RUN AGAIN, ANY OTHER NUMBER TO QUIT: ',$)

C    ------------------------------------------------
C    READ THE NAME OF THE INPUT FILE:
        WRITE (6,10)
        READ  (5,20) INPUT_FILE

C    OPEN INPUT FILE FOR READING

     OPEN  (UNIT=2,ACCESS='DIRECT',FORM='UNFORMATTED',
     2 STATUS='OLD',FILE=INPUT_FILE)
C    ------------------------------------------------

100  CONTINUE

     KEEP_READING = .TRUE.
     DO COUNTER1=1,8,1
        DATA_IN_CHANNEL(COUNTER1)=0
     END DO
     CHANNELS_ENTERED = 0

C    READ WHICH CHANNELS TO PROCESS

     WRITE(6,50)
     DO WHILE (KEEP_READING)
        WRITE(6,60)
        READ(5,70)ACCEPT_CHANNEL

        IF ((ACCEPT_CHANNEL .EQ. -1).AND.
     2  (CHANNELS_ENTERED .NE. 0))THEN
           KEEP_READING=.FALSE.
        END IF

        IF (ACCEPT_CHANNEL .EQ. 0)THEN
           DO COUNTER1=1,8,1
              DATA_IN_CHANNEL(COUNTER1)=1
           END DO
           KEEP_READING=.FALSE.
        END IF

        IF ((ACCEPT_CHANNEL .GE. 1).AND.
     2  (ACCEPT_CHANNEL .LE. 8)) THEN
           CHANNELS_ENTERED=CHANNELS_ENTERED+1-DATA_IN_CHANNEL
```

```
2     (ACCEPT_CHANNEL)
      DATA_IN_CHANNEL(ACCEPT_CHANNEL)=1
   END IF
END DO

C     READ THE NUMBERS OF THE FIRST AND LAST RECORDS TO BE PROCESSED.

      FIRST_RECORD=2
      LAST_RECORD=2

      WRITE (6,90)
      READ  (5,80) FIRST_RECORD

      DO WHILE (FIRST_RECORD .LE. 2)
         WRITE (6,90)
         READ  (5,80) FIRST_RECORD
      END DO

      WRITE(6,91)
      READ (5,80)LAST_RECORD

      DO WHILE (LAST_RECORD .LT. FIRST_RECORD)
         WRITE(6,91)
         READ (5,80)LAST_RECORD
      END DO

      START_SAMPLE=0
      END_SAMPLE=0

      WRITE (6,92)
      READ  (5,80) START_SAMPLE

      DO WHILE ((START_SAMPLE .LT. 0)
     2         .OR.
     2         (START_SAMPLE .GT.320))
         WRITE (6,92)
         READ  (5,80) START_SAMPLE
      END DO

      WRITE(6,93)
      READ (5,80)END_SAMPLE

      DO WHILE ((END_SAMPLE .LT. 0)
     2         .OR.
     2         (END_SAMPLE .GT. 320)
     2         .OR.
     2         ((END_SAMPLE .LT. START_SAMPLE)
     2          .AND.
     2          (FIRST_RECORD .EQ. LAST_RECORD)))
         WRITE(6,93)
         READ (5,80)END_SAMPLE
      END DO

      ACTION_CHOICE = 0

      DO WHILE ((ACTION_CHOICE .NE. 1)
     2         .AND.
     2         (ACTION_CHOICE .NE. 2))
         WRITE (6,94)
         READ  (5,80) ACTION_CHOICE
      END DO

      IF (ACTION_CHOICE .EQ. 1)THEN
         CLEAN_THE_DATA = .TRUE.
```

```
      ELSE
         CLEAN_THE_DATA = .FALSE.
      END IF

C     ------------------------------------------------
200   DO CHANNEL_COUNTER1=1,8,1 !LOOPS THROUGH ALL 8
C                               CHANNELS
            IF(DATA_IN_CHANNEL(CHANNEL_COUNTER1) .EQ. NO_DATA) THEN
               GO TO 300
            END IF

            IF(CLEAN_THE_DATA)THEN
               CALL CLEAR_ERROR_FLAG (FIRST_RECORD,
     2                                LAST_RECORD,
     2                                START_SAMPLE,
     2                                END_SAMPLE,
     2                                CHANNEL_COUNTER1)
            ELSE
               CALL SET_ERROR_FLAG (FIRST_RECORD,
     2                              LAST_RECORD,
     2                              START_SAMPLE,
     2                              END_SAMPLE,
     2                              CHANNEL_COUNTER1)
            END IF
300      CONTINUE
      END DO
C     ------------------------------------------------------

      WRITE (6,95)
      READ  (5,80) REPEAT_CODE

      IF (REPEAT_CODE .EQ. 1) THEN
         GOTO 100
      END IF

C     END THE PROGRAM
3000  CONTINUE
      CLOSE(UNIT=2)
      STOP
      END
```

Appendix H

D1prephg and D1prephg_multi

The D1prephg, D1prephg_multi, and D1prephg_percent are slightly modified versions of McCoy's prephg program. A different linear accelerator gain was taken into account. The d1prephg_multi program prints the activity indices with time codes into 6 different files. This makes it a lot easier to export the data for further calculations or for plotting. The d1prephg_percent is just like d1prephg_multi, but it also prints out the number of valid data points. This is usefull when computing the motion index from the activity indices.

Below is the D1prephg_percent program:

```
C                  **********************
C                  *    HISTOGRAM       *
C ****************************************************************
C *                                                              *
C *  THIS PROGRAM PROCESSES 15 Min Blocks of Accumulated Data    *
C *  AND PRODUCES A Histogram FOR DATA GATHERED FROM SPACELAB I  *
C *                                                              *
C *                        BY R.K. MCCOY                         *
C *                                                              *
C *                     REV: 190, 4 Jan 85                       *
C *                                                              *
C ****************************************************************
C
      DIMENSION N(6),ACC(6,1024),ID(120),IERR(6),ITE(6),NSA(6),
     *  IDAT(131),A(6,1024),SD(6),RL(6),RU(6),RM(6),PE(6),NTSA(6)
      DOUBLE PRECISION SUMX(6),SUMXSQ(6),SX(6),SXSQ(6)
      CHARACTER*15 NAME,RA,CH
      CHARACTER P(131)
      INTEGER PRINTOUT_COUNTER

C
C   ARRAYS INDICATING THE CHANNEL NUMBERS
C
      DATA N/1,2,3,5,7,8/
C
```

```
C Input/Output Format Statements
C
5       FORMAT(/,' NEGATIVE NUMBER UNDER SQUARE ROOT',I1,F8.4)
10      FORMAT(1A)
15      FORMAT( '1',T35,'HISTOGRAM For ',A15,' (Channel ',I1,')',
     *  '  From Tape ',A15,/,T30,'Enlargement Factors:  Horizontal  = ',
     *  F8.3,'; Vertical = ',F8.3,/,T50,'From CDTR Time ',I2,':',I2,
     *  ' To ',I2,':',I2,/,1X,'Max Percentage =',F4.1)
16      FORMAT( 1X,131A1)
17      FORMAT( 1X,131A1,/,1X,F6.1,T129,F5.1,/,T67,
     *  A15,/,T65,A15,/,T10,'Data Mean = ',F7.3,1X,A15,
     *  ' Standard Dev = ',F7.3,'  (Confidence Interval = ',F7.3,
     *  ' to ',F7.3,')',/,' Number of Data Values Not Included',
     *  ' in Graph = ',I8,'  (Percentage = ',F8.5,')')
20      FORMAT(/,' INPUT FILE SPEC: ',$)
22      FORMAT(1X,'Do you want a summary of record information? (Y/N)',$)
25      FORMAT(/,' PLEASE ENTER YES OR NO ',$)
30      FORMAT(1X,'Record# ',I2,3X,'Start - End Times ',I2,':',I2,' - ',
     *  I2,':',I2,3X,'Count = ',I5)
35      FORMAT(1X,'CHANNEL',3X,'ERRORS',3X,'%ERRORS',3X,'SAMPLES',5X,
     *  'MEAN',3X,'STAND. DEV.',3X,'CONFIDENCE INTERVAL:  LOWER - UPPER',
     *  6(/,T4,I1,T12,I6,T21,F5.1,T31,I5,T40,F7.2,T51,F6.2,T84,F6.2,
     *  ' - ',F6.2))
40      FORMAT(/,1X,'Please, Enter the Desired Record #(s) Followed by'
     *  ,'a CR ;  Enter a CR w/o a Number When All Records Have Been ',
     *  'Entered ')
42      FORMAT(1X,'Record #? ',$)
45      FORMAT(I3)
50      FORMAT(1X,'No Records Selected;  Enter 1, if Correct ',$)
55      FORMAT(1X,I3,' Records Selected.  Do You Wish to Reenter Record ',
     *  '#(s);  Please Enter Yes or No ',$)
60      FORMAT(1X,'I/O Error:  Record ',I2,' Does Not Exist on This '
     *  'File;  Enter 1 to Reenter Record #(s) or Zero to Continue ',$)
65      FORMAT(1X,'Record ',I2,' Not Included')
70      FORMAT(1X,'Enter Enlargement Factors',/,1X,'Enlargement factors',
     *  ' allow the adjustment of both horizontal and vertical scales.',
     *  /,1X,'(For default of 2, enter <CR> for each.)',/,1X,
     *  'Horizontal Factor? ',$)
75      FORMAT(1X,'Vertical Factor? ',$)
76      FORMAT(' ',F10.7,'      ',F5.2)
80      FORMAT(1X,'Horizontal Factor = ',F8.3,';  Vertical Factor = ',
     *  F8.3,/,1X,'Is this correct? (Y/N) ',$)
85      FORMAT(F8.3)
94      FORMAT(/,' RUN ANOTHER? (PLEASE ENTER YES OR NO) ',$)
C-------------------------------------------------------------
        OPEN(UNIT=11,STATUS='NEW',FILE='CHANNEL1.DAT')
        OPEN(UNIT=12,STATUS='NEW',FILE='CHANNEL2.DAT')
        OPEN(UNIT=13,STATUS='NEW',FILE='CHANNEL3.DAT')
        OPEN(UNIT=14,STATUS='NEW',FILE='CHANNEL5.DAT')
        OPEN(UNIT=15,STATUS='NEW',FILE='CHANNEL7.DAT')
        OPEN(UNIT=16,STATUS='NEW',FILE='CHANNEL8.DAT')
C-------------------------------------------------------------

C
C ***************************************************************
C *               READ USER SUPPLIED INFORMATION                *
C ***************************************************************
C
C READ THE NAME OF INPUT FILE
C
99      WRITE(6,20)
        READ(5,10)NAME
        OPEN(UNIT=1,ACCESS='DIRECT',FORM='UNFORMATTED',
     *  STATUS='OLD',READONLY,FILE=NAME)
```

```
C
C Histograms only?
C
      WRITE(6,22)
      READ(5,10)RA
      IF(RA.EQ.'N'.OR.RA.EQ.'NO')GO TO 100
C
C READ TIMES
C (IHS = START HOUR, IMS = START MINUTE, IHE = END HOUR,
C IME = END MINUTE, KNT = DATA COUNT, PE = PERCENT ERROR)
C
      DO 95 I=1,120
        READ(1,REC=I,ERR=100)IHS,IMS,IHE,IME,KNT,
     *    (NSA(K),SX(K),SXSQ(K),(A(K,L),L=1,1024),K=1,6)
        XKNT=FLOAT(KNT)
        DO 96 M=1,6
          SAMP=FLOAT(NSA(M))
          IERR(M)=KNT-NSA(M)
          ERR=FLOAT(IERR(M))
          PE(M)=ERR/XKNT
          IF(NSA(M).EQ.0)THEN
            RM(M)=512.
            SD(M)=0.
            RL(M)=512.
            RU(M)=512.
            GO TO 96
          END IF
          RM(M)=SX(M)/SAMP
C         THE FOLOWING IS THE ORIGINAL LINE
C         SD(M)=SQRT(SXSQ(M)/SAMP-RM(M)**2)
          IF ((SXSQ(M)/SAMP-RM(M)**2) .LT. 0.0) THEN
             WRITE(6,5)M,(SXSQ(M)/SAMP-RM(M)**2)
             SD(M)=SQRT(-1.0*(SXSQ(M)/SAMP-RM(M)**2))
          ELSE
             SD(M)=SQRT(SXSQ(M)/SAMP-RM(M)**2)
          END IF
          RL(M)=SD(M)*SQRT(1-PE(M))
          RU(M)=SD(M)*SQRT(1+3*PE(N))
96        PE(M)=PE(M)*100.
C
C Scale Statistical Values
C
C       THE FOLLOWING LINE IS FROM THE ORIGINAL MCCOY'S PROGRAM
C         R=19./512.
C       THE FOLLOWING LINE IS A SUBSTITUTE TO CORRECT FOR
C       DIFFERENT GAIN ON NEW LINEAR ACCELEROMETERS
          R=5./512.
          DO 97 M=1,3
            RM(M)=(RM(M)-512.)*R
            RL(M)=RL(M)*R
            RU(M)=RU(M)*R
97          SD(M)=SD(M)*R
          R=150./512.
          DO 98 M=4,6
            RM(M)=(RM(M)-512.)*R
            RL(M)=RL(M)*R
            RU(M)=RU(M)*R
98          SD(M)=SD(M)*R
c       THE FOLLOWING LINES ARE FROM THE ORIGINAL RKM PROGRAM:
C         WRITE(6,30)I,IHS,IMS,IHE,IME,KNT
C         WRITE(6,35)(N(M),IERR(M),PE(M),NSA(M),RM(M),SD(M),RL(M),RU(M),M=1,6)
C       THE FOLLOWING ARE THE SUBSTITUTES FOR THE ORIGINAL WRITES:
          DO PRINTOUT_COUNTER=1,6,1
            WRITE(10+PRINTOUT_COUNTER,76)(IHS*1.0+IMS/60.0),
```

```
     2         SD(PRINTOUT_COUNTER)
       END DO
95     CONTINUE
C
C Enter The Desired Records
C
100    NREC=0
       WRITE(6,40)
       DO 105 I=1,120
         WRITE(5,42)
         READ(5,45)ID(I)
         IF(ID(I).EQ.0)GO TO 110
         NREC=NREC+1
105    CONTINUE
C
C If No Records Have Been Entered, Determine if Correct
C
110    IF(NREC.EQ.0)THEN
         WRITE(6,50)
         READ(5,45)IQUIT
         IF(IQUIT.EQ.1)GO TO 1000
         GO TO 100
       END IF
C
C Determine if the User Wishes to Reenter Record #s
C
       WRITE(6,55)NREC
       READ(5,10)RA
       IF(RA.EQ.'Y'.OR.RA.EQ.'YES')GO TO 100
C
C Enter Enlargement Factors
C
112    WRITE(6,70)
       READ(5,85)HENLG
       WRITE(6,75)
       READ(5,85)VENLG
       IF(HENLG.LT.1)HENLG=2.
       IF(VENLG.LT.1)VENLG=2.
       WRITE(6,80)HENLG,VENLG
       READ(5,10)RA
       IF(RA.EQ.'Y'.OR.RA.EQ.'YES')GO TO 113
       GO TO 112
C
C Initialize
C
113    IKNT=0.
       DO 120 K=1,6
         SUMX(K)=0.DO
         SUMXSQ(K)=0.DO
         NTSA(K)=0
         ITE(K)=0
         DO 115 L=1,1024
115        ACC(K,L)=0.
120    CONTINUE
C
C Read all Desired Data
C
       DO 140 J=1,NREC
         I=ID(J)
         READ(1,REC=I,ERR=135)IHS,IMS,IHE,IME,KNT,
     *   (NSA(K),SX(K),SXSQ(K),(A(K,L),L=1,1024),K=1,6)
         XKNT=KNT
C
C If the First Record, Find Beginning Time
```

```
C
          IF(J.EQ.1)THEN
            ITHS=IHS
            ITMS=IMS
          END IF
C
C Find Ending Time
C
          IF(J.EQ.NREC)THEN
            ITHE=IHE
            ITME=IME
          END IF
C
C Accumulate Data
C
            TKNT=TKNT+KNT
          DO 130 K=1,6
            NTSA(K)=NTSA(K)+NSA(K)
            SUMX(K)=SUMX(K)+SX(K)
            SUMXSQ(K)=SUMXSQ(K)+SXSQ(K)
            ITE(K)=ITE(K)+IERR(K)
            DO 125 L=1,1024
125           ACC(K,L)=ACC(K,L)+A(K,L)
130       CONTINUE
          GO TO 140
C
C If Read Error Occurs, Inform User Record # Not Included
C
135       WRITE(6,65)ID(J)
140     CONTINUE
C
C ****************************************
C *      Write Output to Terminal        *
C ****************************************
C               * Histograms *
C               **************
C
500     DO 1000 L=1,6
          ERR=FLOAT(ITE(L))
          SAMP=FLOAT(NTSA(L))
          IF(SAMP.EQ.0)GO TO 1000
          ICHAN=N(L)
C
C Calculate Data Statistics (Upper SDEV range calculated
C using 2*SDEV as upper range of data during error.)
C
          RMEAN=SUMX(L)/SAMP
          VAR=SUMXSQ(L)/SAMP-RMEAN*RMEAN
          SDEV=SQRT(VAR)
          PE(L)=ERR/TKNT
          CIL=SDEV*SQRT(1-PE(L))
          CIU=SDEV*SQRT(1+3*PE(L))
C
C Scale Histogram For +/- (1/HENLG) of Maximum Range From
C   the Mean to be Displayed in 131 Columns
C
          ILO=INT(RMEAN-512./HENLG+.5)
          IF(ILO.LT.1)ILO=1
          IHI=INT(RMEAN+512./HENLG+.5)
          IF(IHI.GT.1024)IHI=1024
          STEP=1024./HENLG/131.
C If STEP is Greater Than 1, More Than One Data Values Must
C         Represented by Each Graphical Value
C
```

```
          ILS=ILO
          IF(STEP.GE.1.)THEN
            DO 515 I=1,131
              IDAT(I)=0
              INC=ILO+INT(I*STEP)
510           IDAT(I)=IDAT(I)+ACC(L,ILS)
              ILS=ILS+1
              IF(ILS.LT.INC)GO TO 510
515         CONTINUE
C
C If STEP is Less Than 1, Some Data Values Must be Graphed
C          Over More Than One Graphical Value
C
          ELSE
            DO 520 I=1,131
              INC=ILO+INT(I*STEP)
              IF(ILS.LT.INC)ILS=ILS+1
520           IDAT(I)=ACC(L,ILS)
          END IF
C
C Find Percentage of Most Frequent Acceleration
C
          RMAX=IDAT(1)
          DO 535 I=1,131
          IF(IDAT(I).GE.RMAX)RMAX=IDAT(I)
535       CONTINUE
          PCMAX=FLOAT(INT(RMAX/SAMP*1000.+.5))/10.
C
C Write Heading and Label For Individual Channel Data
C
          GO TO(522,524,526,526,530,530,532,534)ICHAN
522        CH='X Linear Acc'
          GO TO 528
524        CH='Y Linear Acc'
          GO TO 528
526        CH='Z Linear Acc'
528        RANGE=19.
           RA='Gs'
          GO TO 540
530        CH='Roll Angular Acc'
          GO TO 536
532        CH='Pitch Ang. Acc'
          GO TO 536
534        CH='Yaw Ang. Acc'
536        RANGE=150.
           RA='Rad/sec**2'
540       WRITE(6,15)CH,ICHAN,NAME,HENLG,VENLG,ITHS,
     *     ITMS,ITHE,ITME,PCMAX
C
C Scale Frequencies and Draw Body of Graph (Height = 70 Lines)
C
          DO 550 I=1,131
            P(I)=' '
550         IDAT(I)=INT(VENLG*IDAT(I)*70/SAMP+.99)
          DO 570 J=70,1,-1
            DO 560 I=1,131
              IF(IDAT(I).GE.FLOAT(J))P(I)='#'
570         WRITE(6,16)P
C
C Label Data Axis With Data Range and Provide Statistics in
C Actual Data Values  (Data Value 512 = Actual Value 0)
C
          R=RANGE/512.
          RMEAN=(RMEAN-512.)*R
```

```
          SDEV=SDEV*R
          SDEV3=SDEV*3.
          CIU=CIU*R
          CIL=CIL*R
          XLO=(ILO-512.)*R
          XHI=(IHI-512.)*R
          NOT=0
          DO 580 I=ILO,IHI
580         NOT=NOT+ACC(L,I)
          NOT=SAMP-NOT
          PCNOT=FLOAT(NOT)/SAMP*100
          DO 600 I=1,131
600         P(I)='-'
          P(66)='+'              ! Mark Mean Location
          WRITE(6,17)P,XLO,XHI,RA,CH,RMEAN,RA,SDEV,CIL,CIU,NOT,PCNOT
1000    CONTINUE
C
C RUN ANOTHER?
C
        WRITE(6,94)
1005    READ(5,10)RA
        IF(RA.EQ.'Y'.OR.RA.EQ.'YES')GO TO 99
        IF(RA.EQ.'N'.OR.RA.EQ.'NO')GO TO 1010
        WRITE(6,25)
        GO TO 1005
C
C END PROGRAM
C
1010    CLOSE(UNIT=1)
        CLOSE(UNIT=11)
        CLOSE(UNIT=12)
        CLOSE(UNIT=13)
        CLOSE(UNIT=14)
        CLOSE(UNIT=15)
        CLOSE(UNIT=16)
        STOP
        END
```

Appendix I

Sample File Chop

```
      logical*2 input(2560)

      open (unit=1,
     2       access='direct',
     2       form='unformatted',
     2       status='old',
     2       readonly,
     2       file= 'd1SN1094CLEAN.dat')
      open (unit=2,
     2       access='direct',
     2       form='unformatted',
     2       status='new',
     2       recl=1280,
     2       file= 'd1SN1094short.dat')
      do 100 I=1,4
      read (1,rec=I+5,err=1000) input
      write (2,rec=I,err=2000) input
100   continue
1000  continue
2000  continue
      close (unit=2)
      close (unit=1)
      stop
      end
```

Bibliography

[1] Robert K. McCoy. *1985 The Study Of Human Head Movement On Spacelab1.* SM Thesis, MIT, Dept. of Aeronautics and Astronautics.

[2] Oman C.M., Kenyon R.V., Lichtenberg B.K., Money K.E., Watt D.G.D., Young L.R. *1984 Etiological Mechanisms of Space Motion Sickness.* Science Investigation and Technical Plan, Vol.I, Proposal for National Aeronautics and Space Administration, Center for Space Research, MIT, Cambridge, MA.

[3] Oman C.M., Lichtenberg B.K., and Money K.E. *Space Motion Sickness Monitoring Experiment: Spacelab 1, NATO-AGARD Aerospace Medical Panel Symposium on Motion Sickness: Mechanisms, Predictions, Prevention and Treatment, Paper 35, Williamsburg, VA.*

[4] E.I. Matsnev *Report presented at 12th Joint Soviet American Working Group on Space Biology and Medicine, Washington, DC, November 12, 1981*

[5] Davis J.R., Vanderploeg J.M., Santy P.A., Jennings R.T., Stewart D.F. *Space Motion Sickness During 24 Flights of the Space Shuttle* Aviation Space and Environmental Medicine, 1988.

[6] NASA, 1982, *VAX 11/780 SRI playback unit software operation instructions*

[7] Oman C.M., Young L.R., Watt D.G.D, Money K.E., Lichtenberg B.K., Kenyon

R.V., Arrott A.P. *MIT/Canadian Spacelab Experiments on Vestibular Adaptation and Space Motion Sickness* Basic and Applied Aspects of Vestibular Functions J.C. Hwang, N.G. Daunton and V.J. Wilson (Eds.) Hong Kong University Press, Hong Kong, 1988.

[8] Oman C.M., Lichtenberg B.K., Money K.E., McCoy R.K. *MIT/Canadian Vestibular Experiments On the Spacelab-1 Mission: 4. Space Motion Sickness: Symptoms, Stimuli, and Predictability* Experimental Brain Research, 64:316-34

[9] Bock O., Oman C. 1982 *Dynamics of Subjective Discomfort in Motion Sickness as Measured with a Magnitude Estimation Method* Aviation Space and Environmental Medicine p.773-777

[10] Kendall and Stuart *The Advanced Theory of Statistics, Volume I, Forth edition, pp. 90-91* MacMillan NewYork, 1977